Universidad de Alcalá

TRABAJO FIN DE GRADO

Escuela politécnica superior

GRADO EN SISTEMAS DE INFORMACIÓN

DESARROLLO DE UN PORTAL WEB MEDIANTE SPRING 3 Y HTML5

IVÁN RAMIRO COLODRO SABELL

UNIVERSIDAD ALCALÁ DE HENARES

Escuela Politécnica Superior

TRABAJO FIN DE GRADO

DESARROLLO DE UN PORTAL WEB MEDIANTE SPRING 3 Y HTML5

Autor: Iván Ramiro Colodro Sabell

Director/es : Antonio Moratilla Ocaña

TRIBUNAL:

Presidente:_____

Vocal 1º:_____

Vocal 2º:_____

CALIFICACIÓN:_____

FECHA:_____

RESUMEN

La realización de este proyecto fin de grado, pretende por un lado plasmar los conocimientos adquiridos en los cuatro años de estudios en Sistemas de Información, así como en lo posible, tratar de explorar y añadir nuevos conocimientos en materia de desarrollo de aplicaciones WEB, siguiendo las directrices del nuevo modelo de diseño de servicios de información en Internet denominado Web 2.0. Para ello se usarán tecnologías inscritas dentro del marco Web2.0, como Rich Internet Applications (RIA), JSON, JQuery, Ajax, CSS, y HTML5 entre otras.

Web2.0 se gesta en la conferencia sobre la Web2.0 de O'Reilly Media en 2004 y está ligado estrechamente con fundador y presidente de O'Reilly Media, Tim O'Reilly. Se centra en los siguientes conceptos fundamentales:

- Existencia de Web dinámicas que comparten la Información entre sí.
- Facilidad de interoperabilidad del usuario con la aplicación.
- Diseño centrado en el usuario.

También, siguiendo el auge de los dispositivos móviles como vehículo de acceso a internet y sus cada vez mejores prestaciones, se diseñará un portal Web, que buscará la facilidad de uso y efectos visuales llamativos, adaptándose a las características de dichos dispositivos.

Como patrón de diseño para implementar el Sistema se utilizará el conocido Modelo Vista Controlador (MVC), arquitectura estándar para aplicaciones interactivas y que separa los datos y la lógica de negocio de una aplicación de las vistas del usuario. Este patrón de diseño facilita el desarrollo y posterior mantenimiento de las aplicaciones gracias a la reutilización de código y a la separación de conceptos.

Por otro lado, se ha querido construir una aplicación Web que fuera original y útil, cuyos procesos de negocio ayudaran en la medida de lo posible a preservar algo tan importante y a veces tan escaso, como es el agua, dando la oportunidad a los usuarios de la web, tanto públicos como privados, de obtener la previsión del día actual y semanal de riego en función de la situación geográfica del terreno que se pretende regar y del tipo de emisor de riego del que dispongan.

Esto es posible mediante la previsión precisa de la evapotranspiración de referencia ET. La evapotranspiración contabiliza el flujo de agua evaporado en la interfaz Tierra-atmósfera por la tierra y los cuerpos con agua y también por la transpiración de la vegetación a consecuencia de los procesos fotosintéticos. ET es un componente muy importante en el estudio de los ciclos del agua y está asociado al los procesos de intercambio de flujos de calor, que intervienen en los ciclos del agua y energía; en otras palabras, el intercambio de flujos de calor, constituye la energía necesaria para la evapotranspiración. La evaluación de la evapotranspiración en la superficie terrestre resulta de vital importancia para el objetivo del presente proyecto que a resumidas cuentas, sintetizando la idea anterior, es la pérdida de agua en la superficie debido a factores como la temperatura, la radiación solar, precipitaciones, etc. Por tanto, la ET constituye uno de los pilares principales para el cálculo del tiempo de riego óptimo y consecuentemente hacer un uso eficiente del agua. Así pues, con todo lo presentado nace el **proyecto SMART RAIN.**

ÍNDICE

CAPITULO 1. INTRODUCCIÓN

OBJETIVOS

El objetivo principal del trabajo es el desarrollo de un portal Web Siguiendo las directrices Web 2.0. Para ello, se utilizará el lenguaje HTML en su última especificación (HTML5).

En la parte Servidora se utilizará el patrón de diseño MVC implementado con Spring MVC, así como Spring Security para dotar de seguridad de acceso de usuarios. Para el almacenamiento de la Base de datos se utilizará POSTGRESQL e Hibernate como motor de persistencia.

En la parte cliente se usarán las tecnologías apropiadas para crear una aplicación RIA como son la utilización de CSS, JavaScript, JQuery y el marco Bootstrap. Como formato de intercambio de datos entre en Front-end y el Back-end se utilizará JSON.

Para ello, se plantean los siguientes objetivos específicos:

1. Estudiar el desarrollo de aplicaciones RIA, HTML5 JSON, JavaScript, JQuery y Bootstrap en la profundidad necesaria para afrontar el desarrollo de la Web2.0.
2. Estudiar el patrón de diseño MVC implementado mediante Spring MVC.
3. Persistir un modelo de datos mediante POSTGRESQL usando Hibernate.
4. Diseñar una Web que permita a los usuarios registrarse, logearse y así interaccionar con la web utilizando una serie de servicios propios.
5. Producir un manual de usuario detallado sobre la instalación y funcionamiento de la Web.
6. Redactar documentación o memoria final. Cabe destacar la elevada importancia de este punto, ya que la Web generada será objeto de estudio y desarrollo en posteriores Trabajos Fin de Carrera, por lo cual este punto es esencial para facilitar la tarea de estudio y desarrollo de la Aplicación.

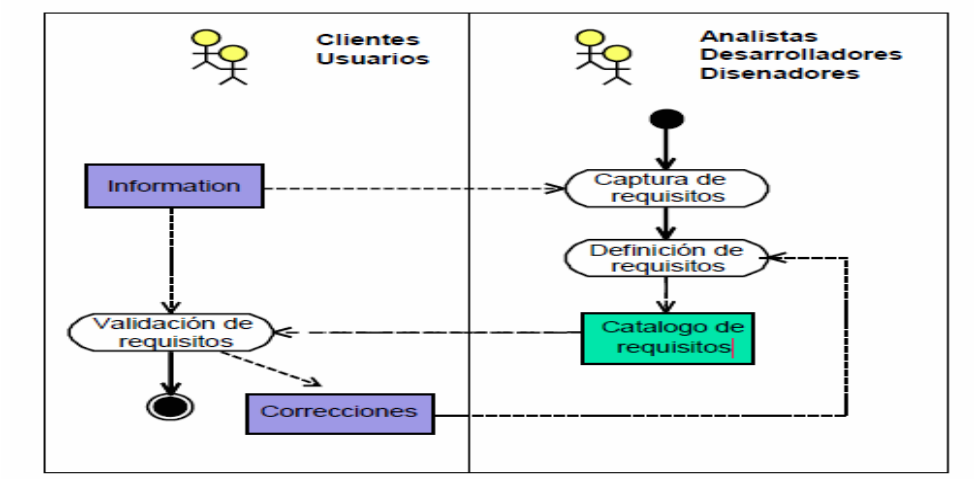
El segundo Objetivo ha sido llevar a cabo la construcción de un portal en la nube, que ayudara a hacer sostenible el consumo de agua en el regadío, ya que el incremento de campos de golf y jardines en España y especialmente en la costa mediterránea ha dejado seco a muchos acuíferos y humedales, como las Tablas de Daimiel. Cabe destacar que el regadío consume el 80 % del agua en España.

ENFOQUE Y METODO A SEGUIR

La mayoría de las propuestas metodológicas para la Web, se han centrado principalmente en las etapas de diseño e implementación, olvidándose de la importancia que tiene la ingeniería de requisitos y el análisis previo antes de acometer las citadas etapas de diseño e implementación.

En este proyecto, se ha dado mucha importancia a la ingeniería de requisitos para facilitar tanto la etapa de diseño como la implementación, logrando un producto totalmente de acuerdo a lo que el usuario quiere.

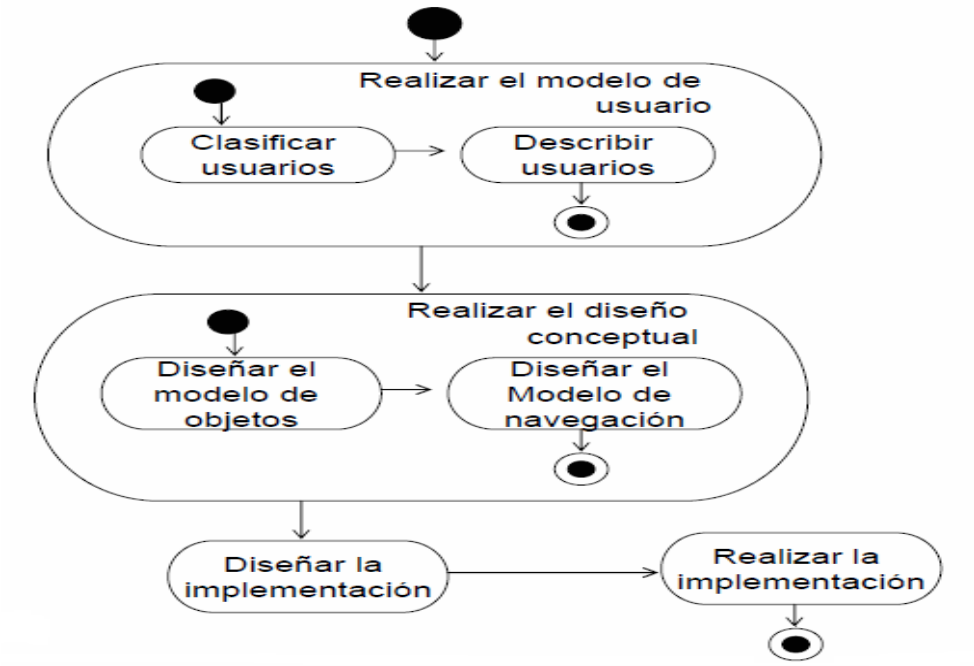
Así podemos ver en el siguiente esquema un resumen de la ingeniería de requisitos seguida:



Una vez hemos obtenido un catalogo de requisitos validado por los usuarios, se ha pasado al análisis y diseño, aplicando una Ingeniería del Software Orientada a Objetos, siguiendo el modelo UML (Unified Modelling Language) en su versión 2.0.

Por último se han abordado las tareas de construcción, pruebas y documentación.

En resumen, el enfoque y método que se ha seguido, lo podemos ver en el siguiente gráfico:



CAPITULO 2. PROCESOS DE NEGOCIO

Básicamente las tareas que se desea implementar se dividen en los siguientes procesos de negocio dependiendo del tipo de usuario que interacciona con la Web.

ACCESO A LA INFORMACIÓN DEL PORTAL

Toda persona que acceda al Portal Smart Rain, tendrá acceso a una información clara y sencilla acerca de los objetivos, funcionalidad y manera de interactuar con el portal. Asimismo se ofrecerán consejos de cómo evitar el derroche de agua y la importancia que tiene el ahorro en el consumo de agua, tanto para el ecosistema como en relación a la economía individual y mundial.

OBTENCION DE LA PREVISIÓN DE REGADIO PARA EL DÍA ACTUAL

Cualquiera que se conecte al Portal Smart Rain podrá obtener la previsión de tiempo de riego del día actual o semanal de acuerdo a su Ciudad o Pueblo y al equipo de riego del que disponga, con sólo introducir la localidad (ciudad o pueblo) donde desea efectuar el riego y el tipo de emisor de que disponga.

OBTENCION DE LA PREVISIÓN DE REGADIO PARA SIETE DIAS

Cualquiera que se conecte al Portal Smart Rain podrá obtener la previsión de tiempo riego del día actual de siete días, de acuerdo a su Ciudad o Pueblo y al equipo de riego del que disponga, con sólo introducir la localidad (ciudad o pueblo) donde desea efectuar el riego y el tipo de emisor de que disponga.

REGISTRO EN LA WEB

Toda persona que acceda al Portal Smart Rain, podrá registrarse pasando a ser usuario registrado del portal y teniendo acceso a la zona Premium. El usuario registrado tendrá la ventaja de que podrá guardar sus datos de localidad y equipo de riego. Así, siempre que acceda al portal, podrá obtener de manera más fácil la previsión de tiempo riego del día actual o de los próximos siete días, sin tener que introducir sus datos de localidad o equipo de riego. Además siempre que lo desee podrá cambiar dichos datos y realizar nuevas previsiones.

Para registrarse tendrá que proporcionar un email válido y una password.

LOGIN

Todo usuario que se haya registrado en el Portal Smart Rain, para identificarse y acceder a la zona Premium, deberá de introducir su email y password para que el Sistema verifique que se trata de un usuario registrado y le redirija a la zona Premium.

El Sistema no dejará a un usuario que no se haya identificado correctamente, acceder a la zona Premium.

GUARDAR CONFIGURACIÓN

Todo usuario registrado podrá guardar su configuración, datos de zona de regadío y emisor de riego en el momento que lo considere necesario, para así obtener automáticamente las previsiones de tiempo de riego actual o de siete días.

CAPITULO 3. DESCRIPCIÓN GENERAL DEL ENTORNO TECNOLÓGICO

METODOLOGÍA DE ANÁLISIS Y DISEÑO

Para la resolución del problema se va a aplicar una Ingeniería del Software Orientada a Objetos, siendo la metodología que mejor se ajusta a este entorno tecnológico, UML (Unified Modelling Language) en su versión 2.0.

UTILIZACIÓN DE TECNOLOGÍAS BASADAS EN EL CONCEPTO WEB 2.0

En la resolución de la práctica se van a usar determinadas tecnologías que se sitúan dentro del esquema de la WEB2.0, esquema que marca una tendencia hacía la facilidad de interacción del usuario con la interface del lado del cliente (Front-End):

- Dar funcionalidad y aspecto de la plataforma del Web como si fuera software de escritorio.
- Respeto a los estándares del W3C.
- Separación del contenido del diseño con uso de hojas de estilo CSS.
- HTML5.
- Utilización de JavaScript así como el FrameWork Bootstrap del Front-End para mejorar el aspecto de las interfaces del Cliente.
- Ajax (Asincronical javascript and xml).
- JSON para envío de datos al Servidor y recepción de datos en el Cliente.
- Facilitar el posicionamiento con URL sencillos.

J2EE

Para la resolución de la práctica se va a utilizar la tecnología J2EE, ya que se trata de una tecnología Open Source de las que existen gran cantidad de herramientas y Frameworks gratuitos, existiendo también multitud de documentación y APIS de desarrollo muy superior a otras tecnologías como puede ser .NET y esto resulta especialmente útil para agilizar el desarrollo de la práctica.

Entre los motivos por lo que se ha escogido J2EE, están los siguientes:

- Lenguaje JAVA, uno de los más utilizados en el ámbito empresarial.
- Tecnología potente.
- Cubre la mayoría de necesidades tecnológicas.
- Interoperable con otras tecnologías: XML, JavaScript, HTML, etc.,
- Gran cantidad de soporte en la red: APIs, manuales, etc.,
- OpenSource y herramientas de desarrollo gratuitas (Eclipse).
- Muchas utilidades ya creadas y fáciles de integrar.
- Fácil conectividad con Base de Datos: driver JDBC.
- Existencia de Frameworks de desarrollo basados en MVC (objetivo principal de esta práctica).

MODELO DE CAPAS

Para la resolución de la aplicación de la práctica se ha elegido, dentro de la arquitectura cliente/servidor, un modelo de tres capas.

El modelo de capas es una técnica software para separar las diferentes partes de la aplicación, con el objetivo de mejorar su rendimiento, mantenimiento y sus funciones. Esta separación de las diferentes partes hace que los cambios en cualquiera de las capas no afecten o afecten poco a las otras capas en que está dividida la aplicación.

Las tres capas serían las siguientes:

- **Capa de presentación:** Es la capa que ve el usuario, presenta el sistema, captura la información y la presenta la información al usuario en un mismo proceso. Esta capa se comunica únicamente con la capa de negocio.
- **Capa de negocio o lógica:** Se reciben las peticiones del usuario y se envían las respuestas tras el proceso. En esta capa se establecen todas las reglas que deben cumplirse. Se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, también se comunica con la capa de persistencia o de datos para solicitar al gestor de la base de datos para recuperar, modificar o insertar datos en la base de datos.
- **Capa de persistencia o de datos:** Es donde residen los datos y la encargada de acceder a ellos. Está formada por uno o más gestores de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación desde la capa de negocio.

UTILIZACION DEL PATRÓN MVC

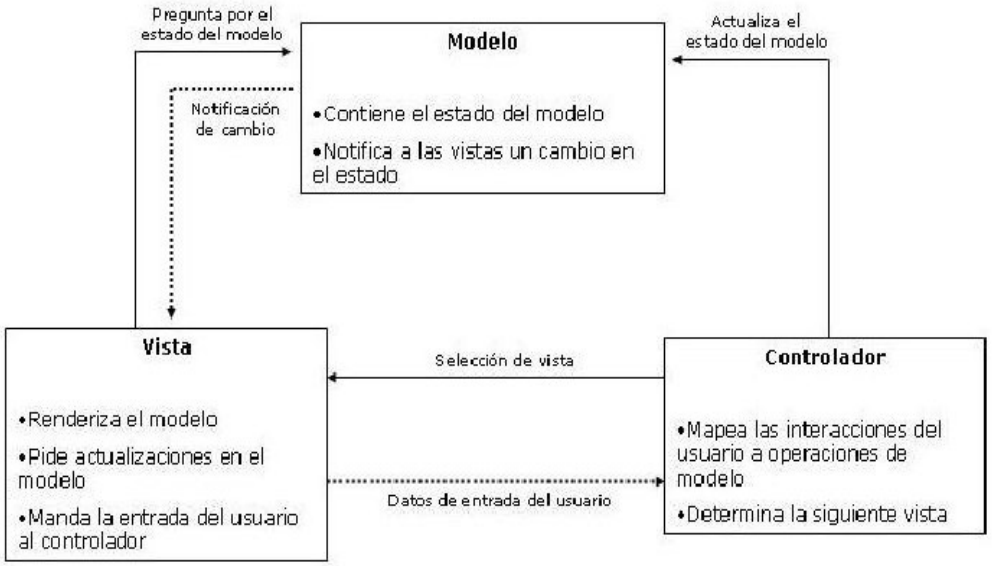
Dentro del modelo de arquitectura en tres capas uno de los patrones de diseño de software más utilizado es el patrón MVC (Modelo, Vista, Controlador)

La evolución de las arquitecturas Web ha sido muy rápida, Podemos considerar como punto clave, la combinación de JSPs y los servlets como implementación del patrón de diseño Modelo-Vista-Controlador al desarrollo Web en Java, lo cual se conoce como Modelo 2 de arquitectura.

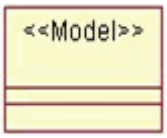
El patrón de diseño Modelo-Vista-Controlador (MVC) fue descrito por primera vez en el año 1979 por Trygve Reenskaug, de los laboratorios de I+D de Xerox. Su característica primordial es la separación de la aplicación en tres capas distintas, el Modelo, la Vista y el Controlador.

Así, el MVC es una guía para el diseño de arquitecturas de aplicaciones que mantienen una gran interactividad con los usuarios. Este patrón organiza la aplicación en tres modelos separados, el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio, el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información y el tercero es un conjunto de controladores que procesan las peticiones de los usuarios y controla el flujo de ejecución del sistema.

El siguiente gráfico básico, nos muestra la interacción entre las tres capas:



Por tanto el MVC divide la arquitectura de una aplicación en 3 tipos de clases fundamentales, con las responsabilidades siguientes:



Clase Modelo <<Model>>:

- Procesa peticiones del sistema.
- Genera datos de respuesta del sistema.
- Gestiona y almacena la información.

Clase Vista <<View>>:

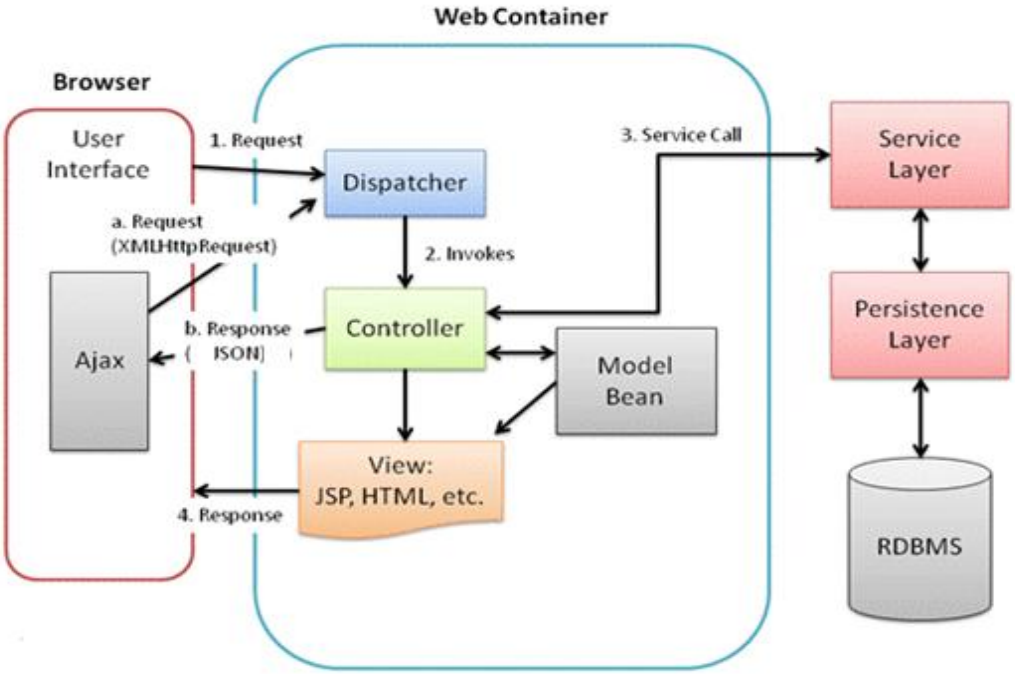
- Genera peticiones.
- Muestra respuestas del sistema.

Clase Controlador <<Control>>:

- Gestiona procesamiento de peticiones.
- Gestiona las respuestas del sistema.

Hay que destacar que el patrón MVC utilizado ha sido extendido debido a la utilización de la tecnología JavaScript y Ajax, para dotar a las interfaces de los usuarios de una mayor experiencia en cuanto a facilidad y usabilidad de las mismas. Además se ha usado un formato especial de transmisión de la información como es el formato JSON:

Por tanto podríamos esquematizar este patrón MVC extendido o mejorado, de la siguiente forma:



Este esquema ilustra un patrón de aplicaciones web usado comúnmente, el cual puede ser tratado como la mejora del tradicional patrón MVC.

1. Request: Una petición es enviada al servidor, la mayoría de los FrameWorks (Spring MVC, Struts, etc.,) tendrán un dispatcher para atender las solicitudes.

2. Invokes: El dispatcher envía las solicitudes al controlador apropiado.

3. Service call: El controlador interactúa con la capa de servicio para hacer uso de la capa de persistencia.

4. Response: El controlador actualiza el modelo basado en el resultado y responde con la correspondiente vista al usuario.

En adición con las llamadas Ajax sucede lo siguiente:

a. Request: Un XMLHttpRequest (Petición Ajax) es preparado y enviado al servidor, el dispatcher enviará la solicitud al correspondiente controlador.

b. Response: El controlador interactúa con la capa de servicio y los datos de respuesta serán formateados y enviados al navegador, en este caso las vistas no son involucradas ya que el navegador recibe los datos y realiza una actualización parcial en la vista existente.

Un ejemplo de uso de esta arquitectura, es el Framework STRUTS que es un proyecto de la "Apache Software Foundation" que tiene como objetivo proporcionar un entorno específico para la construcción de grandes aplicaciones Web. El núcleo del entorno está constituido por un conjunto de tecnologías estándar, como son los Servlets, JavaBeans, ResourceBundles y XML.

Otro ejemplo de Framework que utiliza el patrón MVC es SPRING-MVC, que es un Framework de código abierto de desarrollo de aplicaciones para la plataforma Java. La primera versión fue escrita por Rod Johnson. También hay una versión para la plataforma .NET, Spring.net. Inicialmente el Framework se lanzó bajo la licencia Apache 2.0 en junio de 2003.

FRAMEWORKS UTILIZADOS EN EL DESARROLLO DEL PROYECTO

En general, con el término Framework, nos estamos refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un Framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los Frameworks utilizados para la práctica han sido los siguientes:

- **Framework MVC:**

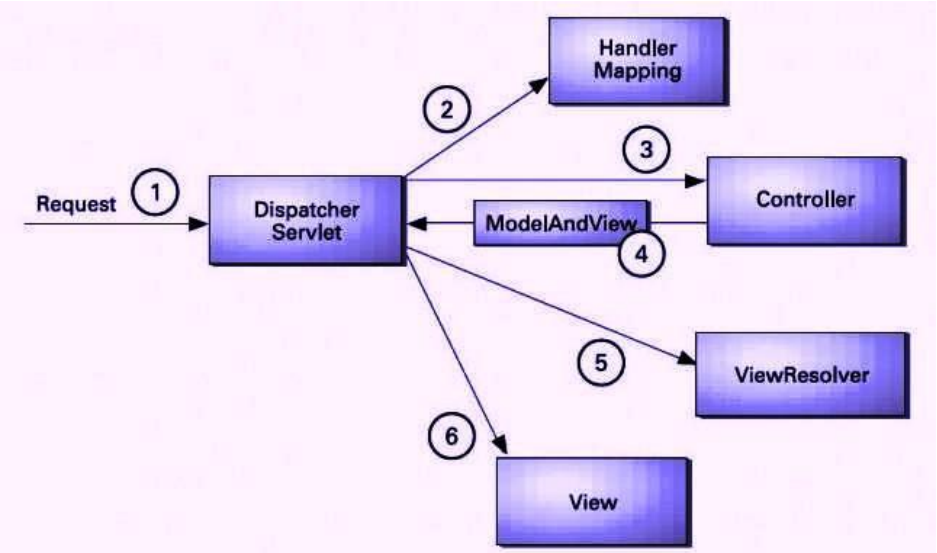
  Para la implementación del patrón MVC se ha elegido Spring MVC frente a Struts-2 que era otra de las opciones tomadas en cuenta para la implementación del patrón MVC de la práctica.

Spring MVC es uno de uno de los módulos del Framework de Spring, y como su propio nombre nos indica que implementa una arquitectura Modelo - Vista - Controlador.

Spring brinda un MVC para Web bastante flexible y configurable, pero esto no le quita sencillez ya que se pueden realizar aplicaciones sencillas sin tener que configurar muchas opciones.

El FrameWork tiene un conjunto de interfaces que después se implementan para proporcionar la funcionalidad correspondiente. Las interfaces están acopladas claramente al Servlet Api.

La clase DispatcherServlet está en el front controller y es responsable de delegar y coordinar el control entre varias interfaces en la fase de ejecución durante una petición Http.



El navegador envía una petición y el distribuidor de servlets se encarga de recoger esta petición (paso 1) para pasárselo al controlador de mapeos (paso 2) que comprueba que dicha petición este mapeada y le devuelve el controlador asociado a dicha petición al distribuidor de servlets. Una vez que sabemos que controlador necesitamos el distribuidor de servlets le pasara el control a dicho controlador (paso 3) para que este se encargue de realizar toda la lógica de negocio de nuestra aplicación, este controlador devolverá un objeto modelo y vista (paso 4), el modelo son la información que deseamos mostrar y la vista donde deseamos mostrar dicha información.

Una vez el distribuidor de servlets tiene el objeto modelo y vista tendrá que asociar el nombre de la vista retornado por el controlador con una vista concreta es decir una página jsp, jsf,. (Paso 5). Una vez resulto esto nuestro distribuidor de servlets tendrá que servir la vista Una vez resulto esto nuestro distribuidor de servlets tendrá que pasar a la vista el modelo, es decir los datos a presentar, y mostrar la vista (paso 6).

VENTAJAS DE SPRING FRENTE A STRUTS

- Spring MVC ofrece una división limpia entre Controllers, Models (JavaBeans) y Views.
- Spring MVC es muy flexible, ya que implementa toda su estructura mediante interfaces, no como Struts que obliga a heredar de clases concretas tanto en sus Actions como en sus Forms.
- Spring MVC provee interceptores y controllers que permiten interpretar y adaptar el comportamiento común en el manejo de múltiples request.
- Los controllers de Spring MVC se configuran mediante IoC como los demás objetos, lo cual los hace fácilmente testeables e integrables con otros objetos que estén en el contexto de Spring, y por tanto sean manejables por éste.
- Las partes de Spring MVC son más fácilmente testeables que las de Struts, debido a que evita la herencia de una clase de manera forzosa y una dependencia directa en el controller del servlet que despacha las peticiones.
- No existen ActionForms, se enlaza directamente con los beans de negocio.
- Struts obliga a extender la clase Action, mientras que Spring MVC no, aunque proporciona una serie de implementaciones de Controllers para que el usuario los utilice. Existen una gran variedad de Controladores.
- Spring tiene una interfaz bien definida para la capa de negocio.

Para crear los controladores seguiremos los siguientes pasos:

1. Definir una clase que implementa la interfaz del controlador.
2. Insertar dicha clase como un objeto en el contexto de Spring.
3. Asignar el nombre a una Uri que posteriormente será invocada por el usuario.
4. Especificar la extensión de la Uri en el DispatcherServlet de Spring, configurado en web.xml, para que se pueda buscar internamente.

TECONLOGÍA JSP PARA CREAR LAS VISTAS

El uso de la tecnología Jsp sería la siguiente:

Tendríamos una clase controlador, que ejecutaría una lógica y devolvería un objeto ModelAndView, con un nombre.

Esta petición pasaría a ser resulta por Spring mediante el bean de tipo ViewResolver, que añadiría un prefijo y un sufijo a dicha jsp.

Cabe destacar, que debido al uso de la tecnológica JavaScript para dotar de funcionalidad a la Vista, al uso de AJAX para las peticiones al Servidor y a la utilización de la transmisión de datos (Modelo) entre el Cliente y el Servidor mediante el formato ligero JSON, una vez que ha sido creada y descargada en el cliente la vista principal de la Web, dicha vista sólo se actualizará en el cliente, mejorando notablemente la experiencia que el usuario tendrá de la aplicación, resultando muy parecida a la de una aplicación de escritorio, no teniendo que descargarse las actualizaciones de las vistas desde el Servidor y por tanto, no produciéndose demoras entre los eventos de los usuarios y el resultado esperado.

El modelo se pasa mediante un atributo de la clase ModelAndView, para posteriormente acceder al mismo desde la parte de la vista.

CONTROL DE ACCESOS

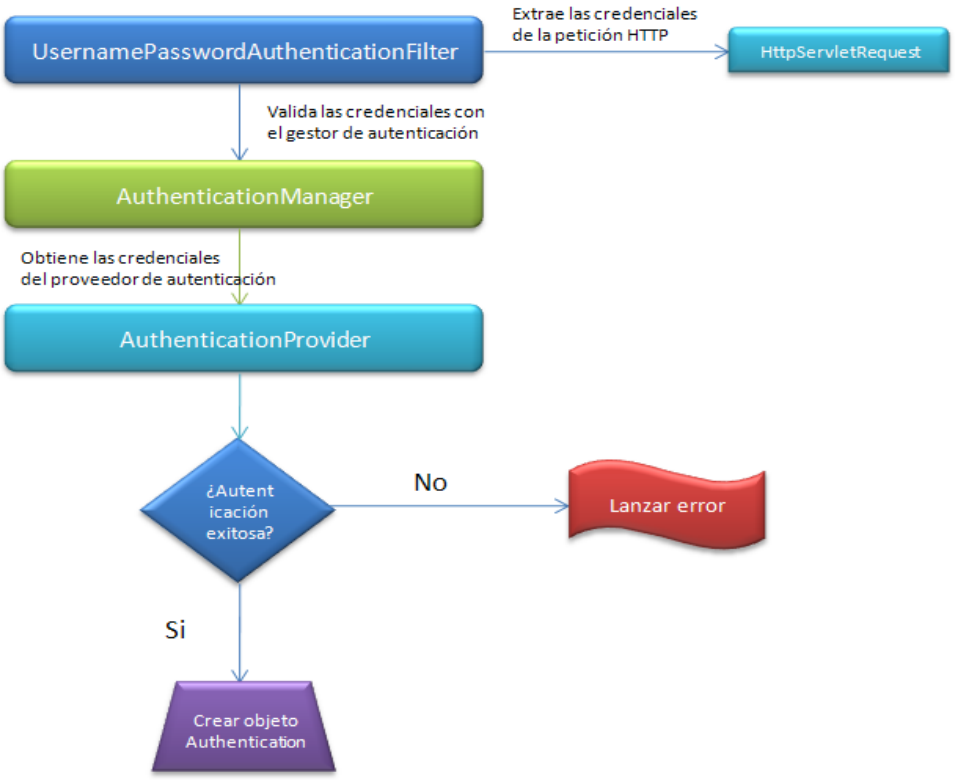
Para controlar los accesos a las vistas de la aplicación reservadas a los administradores del Sistema, se ha añadido a Spring MVC el módulo Spring Security.

Spring Security proporciona servicios de seguridad para aplicaciones de software empresariales basados en J2EE.

Los procesos de seguridad están destinados principalmente, a comprobar la identidad del usuario mediante la autenticación y los permisos asociados al mismo mediante la autorización. La autorización es dependiente de la autenticación ya que se produce posteriormente a su proceso.

Por regla general muchos de estos modelos de autenticación son proporcionados por terceros o son desarrollados por estándares importantes como el IETF adicionalmente, Spring Security proporciona su propio conjunto de características de autenticación.

Un esquema de cómo funciona Spring Security, podría ser el siguiente:



FRAMEWORK ORM

El patrón DAO (Data Acces Object) es uno de los módulos más importantes y utilizados en J2EE y que se usará en el desarrollo de este proyecto.

Hay dos maneras de llevar a cabo la conexión, manejo y acceso a las bases de datos: utilizar algún, Framework ORM o utilizar simplemente JDBC (Java Database Connectivity).

Dado que se quiere dar mayor soporte y mejor robustez a la aplicación, se ha elegido utilizar el Framework Hibernate como ORM (Object-Relational Mapping).

Hibernate es una librería ORM open-source, que facilita trabajar con bases de datos relacionales. Hace que el desarrollador trabaje con objetos (POJO's), lo que lleva a que el desarrollador se preocupe por el negocio de la aplicación y no del cómo se guardara, recuperará o eliminará la información de la base de datos.

Hibernate funciona asociando a cada tabla de la base de datos un "Plain Old Java Object" (POJO, a veces llamado Plain Ordinary Java Object). Un POJO es similar a una Java Bean, con propiedades accesibles mediante métodos setter y getter.

Utiliza su propio lenguaje de consultas llamado HQL.

Cabe destacar que Hibernate encaja perfectamente con Spring MVC resultando las siguientes ventajas con su utilización:

- Manejo de sesión: Spring hace de manera más sencilla, eficiente y segura que cualquier herramienta ORM, la forma en que se manejan las sesiones.
- Manejo de recursos: se puede manejar la localización y configuración de las fábricas de sesiones de Hibernate o las fuentes de datos de JDBC, por ejemplo haciendo que estos valores sean más fáciles de modificar.
- Manejo de transacciones integrado: se puede utilizar una plantilla de Spring para las diferentes transacciones ORM.
- Envolver excepciones: se pueden envolver todas las excepciones para evitar las declaraciones y los catch en cada segmento de código necesario.

FRAMEWORK BOOTSTRAP

Del lado del cliente, Front-End, se utilizará el Framework Bootstrap.

Bootstrap es un framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript, desarrollado por Twitter.

Su mayor ventaja es que se pueden crear interfaces de usuario que se adaptan a los diversos navegadores actuales. Cuenta con multitud de componentes Web, lo que nos ahorra tiempo en el diseño y construcción de las interfaces del usuario.

Además se integra perfectamente con las principales librerías Javascript, como JQuery, utilizada también en la realización de este proyecto.

Ofrece un diseño sólido usando LESS (Lenguaje de Hojas de Estilo Dinámico) y estándares como CSS3/HTML5.

Funciona con todos los navegadores, incluido Internet Explorer usando HTML Shim para que reconozca los tags HTML5.

Dispone de distintos layout, predefinidos con estructuras fijas a 940 píxeles de distintas columnas.

BASE DE DATOS UTILIZADA

Para la realización del proyecto se ha optado por usar como Base de Datos PostgreSQL que es un SGBD relacional orientado a objetos y libre, publicado bajo la licencia BSD.

Algunas de sus principales características es soportar una alta concurrencia y el poseer una amplia variedad de tipos nativos.

Sus principales ventajas son:

- Seguridad en términos generales.
- Integridad en BD: restricciones en el dominio.
- Integridad referencial.
- Afirmaciones (Assertions).
- Disparadores (Tiggers).
- Autorizaciones.
- Conexión a DBMS.
- Transacciones y respaldos.

SERVIDOR WEB

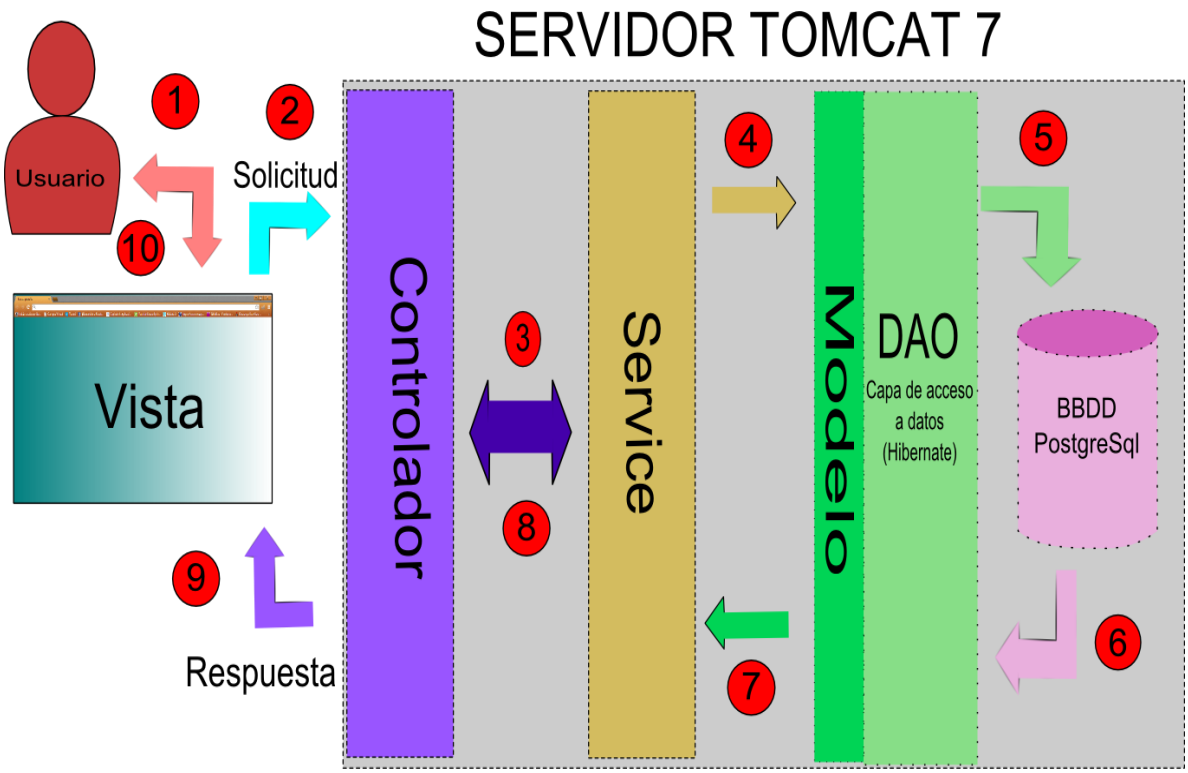
Para la realización del proyecto se ha optado, por su sencillez y extenso uso, a Apache Tomcat como servidor Web. Funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat está escrito en Java y por lo tanto puede funcionar en cualquier Sistema Operativo que disponga de una máquina virtual de Java.

Se ha utilizado la última versión de Tomcat, la 7x, que implementan las especificaciones de Servlet 3.0 y de JSP 2.2. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

Dicho servidor es mantenido en buena parte por la Apache Software Foundation y es de uso gratuito.

ESQUEMA GENERAL



Framework utilizado: Spring 3

ENTORNO DE DESARROLLO

La herramienta IDE seleccionada para la realización de la práctica ha sido Eclipse en su versión Juno, con los siguientes plugins instalados:

- Maven Integration for Eclipse.
- Spring Tool Suite (STS) for Eclipse Juno.

Maven se ha elegido para la gestión y construcción del proyecto, herramienta que se integra perfectamente con el IDE Eclipse.

Spring STS nos ofrece facilidades a la hora de crear un proyecto dinámico Web basado en Spring MVC.

CAPITULO 4. ANALISIS DEL SISTEMA

Este capítulo se va a dedicar a realizar un análisis teórico del Portal Smart Rain desarrollado. Se hace una descripción general del funcionamiento del sistema y a continuación se detallan tanto los requisitos funcionales del sistema como los no funcionales.

DESCRIPCIÓN DEL FUNCIONAMIENTO DEL SISTEMA

El Portal Web desarrollado pretende cubrir las necesidades de los usuarios, tanto públicos como privados, que quieren saber el tiempo que tienen que dedicar al riego del césped de sus jardines o parques, en función del equipo de riego del que dispongan.

El Portal debe de permitir obtener la previsión del día actual y semanal de riego en función de la situación geográfica del terreno que se pretende regar y del tipo de emisor de riego. Esto es posible mediante la previsión precisa de la evapotranspiración de referencia (ET0) de cada zona en particular y mediante la información meteorológica existente en cada país. La evapotranspiración viene a representar un índice climático asociado a una determinada área.

Existen varias metodologías para realizar estas previsiones siendo elegido el método de HARGREAVES, debido a que sólo utiliza información de temperatura media, máxima y mínima del aire y algunos parámetros de ajuste climático para cada zona, como son la radiación .

El método de HARGREAVES es un método relativamente sencillo que se aplica a través de la siguiente ecuación:

$$ET_0 = 0{,}0135\ (t_{med} + 17{,}78)\ R_s$$

Dónde:

ET0: evapotranspiración potencial diaria, mm/día.

Rs: radiación solar incidente, convertida en mm/día, depende entre otros factores, de la latitud y el mes.

Tmed: Temperatura media.

Para la obtención de los datos de temperatura y ajuste climático para poder calcular la evapotranspiración de la zona a regar, se acude a un servidor externo que mediante servicios web y archivos de intercambio JSON, nos ofrece todos los datos climáticos necesarios.

El Portal permite registrarse como Usuario Premium que tiene la ventaja de guardar la configuración del usuario para que cada vez que se conecte, el usuario obtenga su previsión de riego actual o semanal, sin necesidad de volver a indicar la localidad y equipo de riego.

Una vez realizada la descripción del Sistema se analizará más en detalle los requisitos.

Para lo obtención de los requisitos del Sistema se han mantenido entrevistas, tanto con posibles usuarios como con fabricantes y distribuidores de equipos de riego.

Existen dos tipos de requisitos:

- Funcionales. Describen el comportamiento del Sistema, que acciones ha de desarrollar y que resultados debe de ofrecer.
- No Funcionales. Especifican características que ha de cumplir el sistema, en términos de que datos se deben de almacenar, de calidad y de seguridad.

CATALOGO DE REQUISITOS

Identificación de Usuarios

Se han identificado dos clases de Usuarios del Sistema:

Usuario Visitante

Usuarios que acceden al portal "Smart Rain" y acceden a la información sobre el objetivo y funcionamiento del portal y además pueden consultar el tiempo de riego necesario en su localidad y emisor de riego, tanto para la previsión del día actual como semanal. También pueden registrarse en la web y convertirse así en "Usuarios Premium".

Usuario Registrado

Usuarios que al identificarse en el portal como tales, además de tener acceso a las funcionalidades del "Usuario Visitante" pueden guardar su configuración en local y así, siempre que se conecten al portal accederán a las consultas del tiempo de riego de su localidad y equipo de riego actual y semanal, sin necesidad de introducir los datos de su localidad y emisor de riego. También pueden realizar nuevas previsiones cambiando su localidad o equipo de riego y volver a guardar la nueva configuración.

Requisitos Funcionales comunes a Usuarios Visitantes y Usuario Registrado

IDENTIFICADOR	RF-UC001
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier usuario que acceda al Portal Smart Rain, tendrá acceso a la ayuda e información de los contenidos y objetivos del uso del Portal
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RF-UC002
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier usuario que acceda al Portal Smart Rain, podrá consultar el tiempo de riego necesario en el día actual, en base a la localidad donde quiera regar y al emisor de riego del que disponga
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RF-UC003
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier usuario que acceda al Portal Smart Rain, podrá consultar la previsión semanal del tiempo de riego necesario, en base a la localidad donde quiera regar y al emisor de riego del que disponga
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RF-UC004
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier usuario que acceda al Portal Smart Rain, , para poder acceder a las previsiones de tiempo de riego, deberá de introducir la Localidad y la Comunidad Autónoma donde se sitúe el área a regar
PRIORIDAD	Alta
COMENTARIOS	El usuario Visitante, siempre tendrá que introducir dichos datos cuando quiera realizar las previsiones de riego. El Usuario Registrado sólo tendrá que introducirlos cuando quiera modificar los datos de la localidad de riego.

IDENTIFICADOR	RF-UC005
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier usuario que acceda al Portal Smart Rain, para poder acceder a las previsiones de tiempo de riego, deberá de introducir el tipo de emisor con el que realiza el riego, eligiendo el fabricante, el tipo de emisor y las dimensiones de la Tobera/boquilla
PRIORIDAD	Alta
COMENTARIOS	El usuario Visitante, siempre tendrá que introducir dichos datos cuando quiera realizar las previsiones de riego. El Usuario Registrado sólo tendrá que introducirlos cuando quiera modificar los datos del emisor de riego.

Requisitos Funcionales exclusivos de Usuarios Visitantes

IDENTIFICADOR	RF-UV001
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier Usuario Visitante que acceda al portal Smart Rain, podrá acceder a la zona Premium registrándose.
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RF-UV002
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier Usuario Visitante, al registrarse para convertirse en Usuario

	Registrado, deberá de suministrar su dirección de correo electrónico y una contraseña, que le permitirán entrar a la zona Premium. La dirección de correo electrónico deberá de tener un formato válido.
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RF-UV003
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier Usuario Visitante una vez que se haya registrado, accederá automáticamente al Login de entrada como Usuario Registrado.
PRIORIDAD	Alta
COMENTARIOS	

Requisitos Funcionales exclusivos de Usuarios Registrados

IDENTIFICADOR	RF-UR001
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier Usuario Registrado podrá acceder al Login de Entrada a la zona Premium, teniendo que identificarse mediante su dirección de correo electrónico y una contraseña. La dirección de correo electrónico deberá de tener un formato válido.
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RF-UR002
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier Usuario Registrado podrá guardar sus datos de Localidad de Riego y tipo de Emisor utilizado
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RF-UR003
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Funcional
DESCRIPCIÓN	Cualquier Usuario Registrado que haya guardado sus datos de Localidad de Riego y tipo de Emisor utilizado, accederá automáticamente a las previsiones de riego actual y semanal sin necesidad de introducir dichos datos otra vez.
PRIORIDAD	Alta
COMENTARIOS	

Requisitos no Funcionales

IDENTIFICADOR	RNF-001
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Calidad del Sistema
DESCRIPCIÓN	El Sistema deberá implementar la validación de la entrada de datos con la tecnología JavaScript
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RNF-002
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Calidad del Sistema
DESCRIPCIÓN	El Sistema deberá implementar tecnologías inscritas dentro del marco Web2.0, RIA, JSON, JQuery, Ajax, y CSS, y HTML5 entre otras.
PRIORIDAD	Alta
COMENTARIOS	
IDENTIFICADOR	RNF-003
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Calidad del Sistema
DESCRIPCIÓN	Como patrón de diseño para implementar el Sistema se utilizará el conocido Modelo Vista Controlador (MVC)
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RNF-004
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Calidad del Sistema
DESCRIPCIÓN	Como formato para el intercambio de datos entre el modelo de datos alojado en el Servidor y el cliente, se utilizará JSON
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RNF-005
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Calidad del Sistema
DESCRIPCIÓN	E sistema deberá de utilizar la formula de HARGREAVES para el cálculo de la ET para la estimación de los tiempos de riego necesarios para una Localidad determinada.
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RNF-006
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell

TIPO	Seguridad
DESCRIPCIÓN	Los Usuarios Visitantes no podrán acceder a las zona Premium del Portal
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RNF-007
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Seguridad
DESCRIPCIÓN	Los Usuarios Premium deberán de acreditarse con un Nombre de Usuario, que será su dirección de correo electrónico, y una Password, mediante un Login para poder acceder a las vistas del Portal dedicadas a ellos (zona Premium).
PRIORIDAD	Alta
COMENTARIOS	

IDENTIFICADOR	RNF-008
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Seguridad
DESCRIPCIÓN	Las Password de los Usuarios Premium deberán de guardarse y recuperarse codificadas.
PRIORIDAD	Alta
COMENTARIOS	

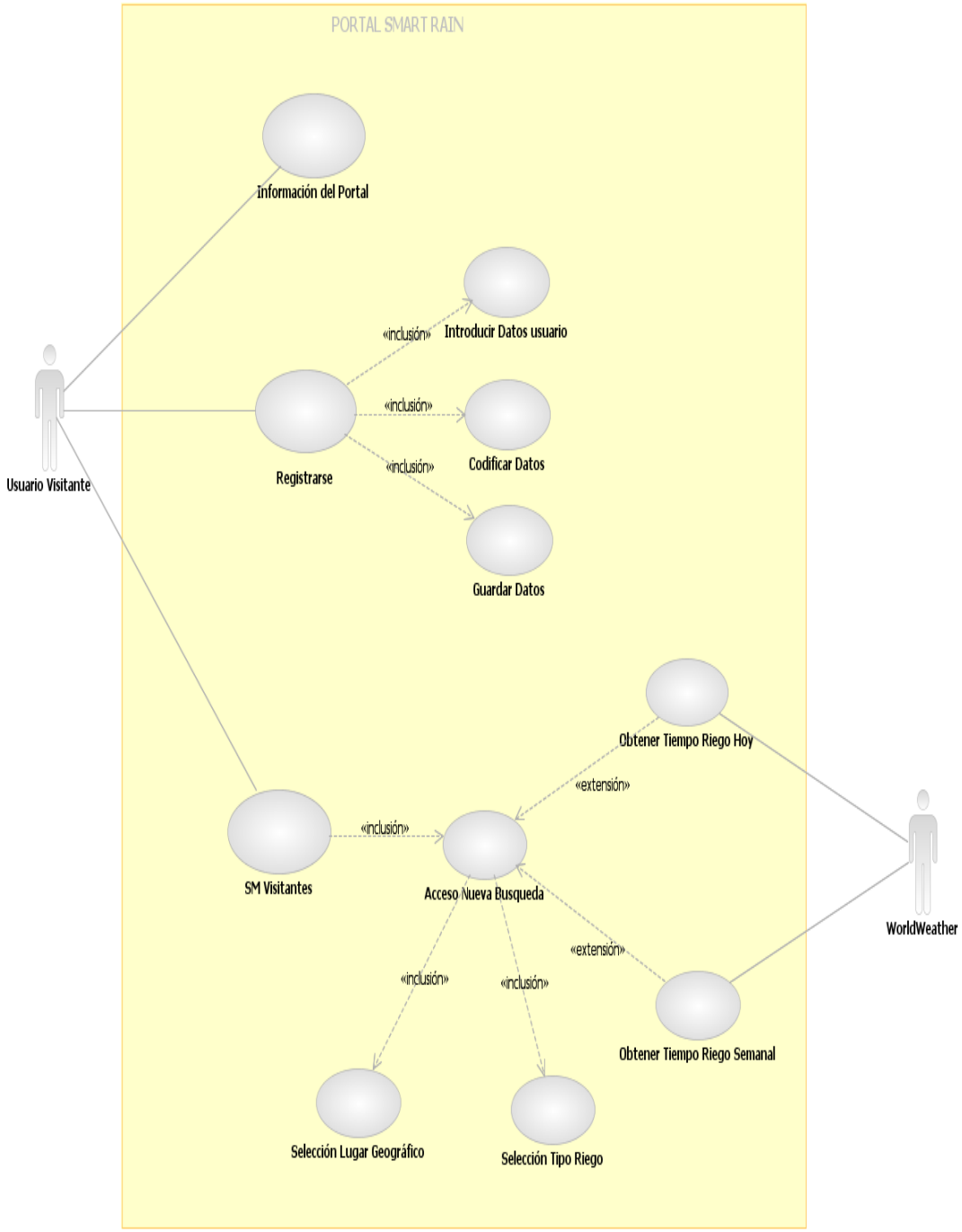
IDENTIFICADOR	RNF-009
VERSIÓN	1.0 (01-04-2013)
AUTOR	Iván Colodro Sabell
TIPO	Sistema
DESCRIPCIÓN	La aplicación debe de suministrarse de datos climatológicos de algún proveedor/empresa que disponga de los mismos.
PRIORIDAD	Alta
COMENTARIOS	

CASOS DE USO

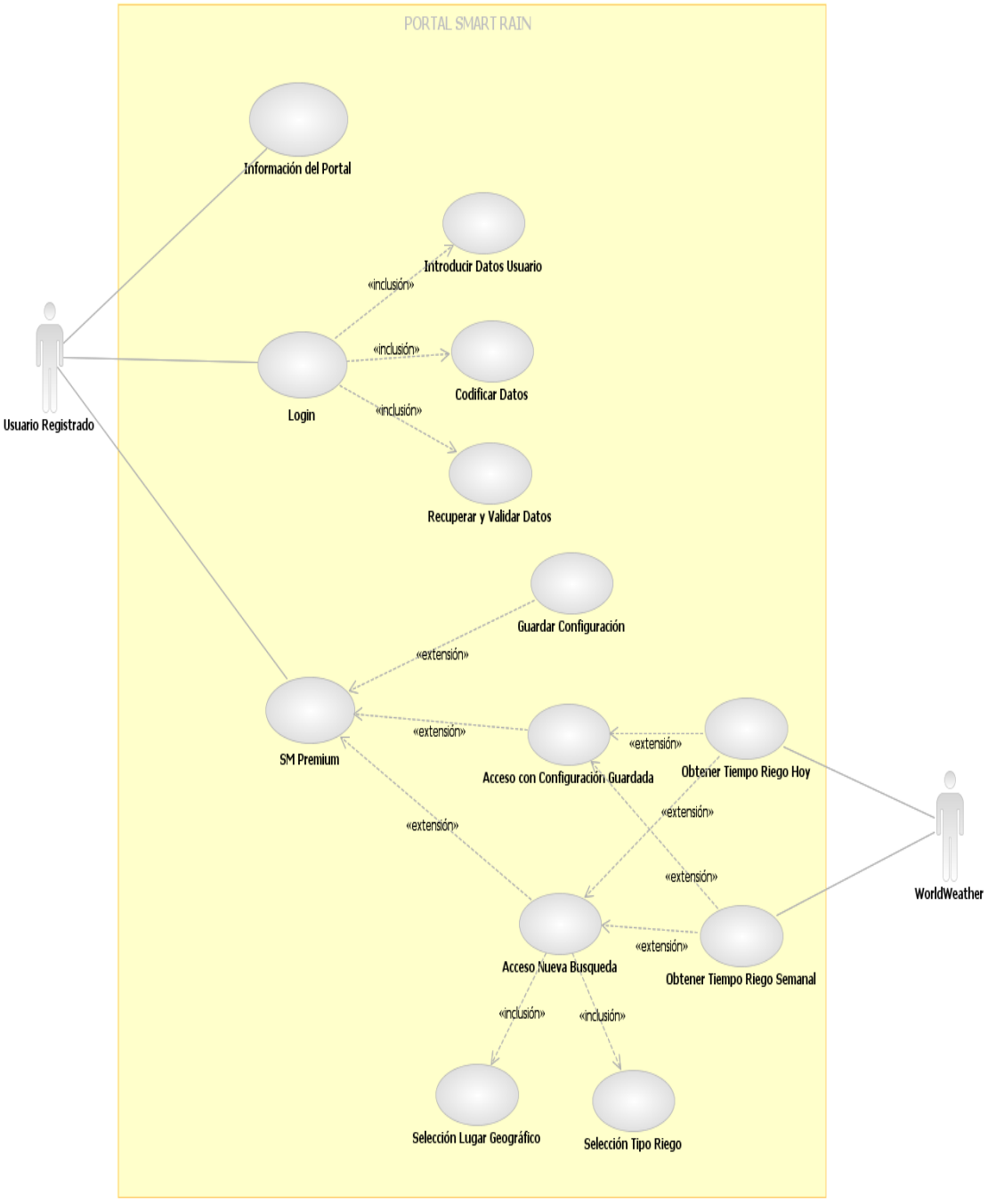
El modelo de casos de uso especifica la funcionalidad que el sistema ha de ofrecer desde la perspectiva de los usuarios y lo que el sistema ha de realizar para satisfacer las peticiones de estos usuarios. Este modelo utiliza tres elementos básicos: actor, caso de uso y relación.

Se han identificado los siguientes Casos de Uso según el tipo de Usuario que accede al Portal Smart Rain:

Usuario Visitante



Usuario Registrado



Hay que destacar que los accesos a los datos climatológicos necesarios se han modelado mediante el actor WorldWeather.

Descripción de los actores

- **Usuario Visitante:** Representa a cualquier usuario que accede al Portal Smart Rain y no está registrado en el Portal.

- **Usuario Registrado: Representa a** cualquier usuario que accede al Portal Smart Rain y está registrado en el Portal.

- **WorldWeather**: Representa al Sistema Externo donde se accede para recopilar los datos climatológicos necesarios, esta empresa ofrece la accesibilidad de datos de las estaciones climatológicas del mundo entre las que se encuentran las españolas. De esta manera es posible obtener los metadatos necesarios para calcular el tiempo de riego para un lugar determinado.

Descripción de los Casos de Uso

- **Información del Portal:** Este Caso de Uso proporciona, tanto el Usuario Visitante como el Usuario Registrado, el acceso a toda la información que se publique en el Portal Smart Rain.

  Requisitos Funcionales que cumple:

  - RF-UC001.

- **Registrarse:** Este Caso de Uso se ocupa del registro de los Usuarios Visitantes para pasar a ingresar como Usuarios Premium. Una vez el Usuario se ha registrado en el Sistema, se le redirige a la vista de Login donde, una vez identificado, accederá a la zona Premium. Incluye los casos de uso:
  - **Introducir Datos Usuario:** Se introduce el Email y Password del Usuario.
  - **Codificar Datos:** Se codifica la Password del Usuario.
  - **Guardar Datos:** Se hacen persistentes en Base de Datos los datos introducidos.

  Requisitos Funcionales que cumple:

  - RF-UV001.
  - RF-UV002.
  - RF-UV003.

- **SMVisitantes:** Este Caso de Uso especifica como un Usuario Visitante puede obtener la previsión de riego del día actual o semanal de acuerdo a su Ciudad o Pueblo y al equipo de riego del que disponga. Incluye el siguiente Caso de Uso:

- **Acceso Nueva Búsqueda:** Encargado de solicitar al Usuario los datos necesarios para efectuar la previsión de Riego solicitada y de efectuar la previsión del día actual o la previsión de siete días. Incluye los siguientes Casos de Uso:

  - **Selección Lugar Geográfico**: Posibilita la entrada de la Ciudad o Pueblo donde se quiere la previsión, siendo necesario introducir la Localidad y la Comunidad Autónoma.

  - **Selección de Tipo de Riego**: Posibilita la entrada del tipo de emisor de riego, siendo necesario introducir el fabricante, el tipo de emisor y las medidas de la tobera o boquilla.

  Extiende los siguientes Casos de Uso:

  - **Obtener Tiempo Riego Hoy:** Este Caso accede al Sistema Externo **"**WoldWeather" para obtener los datos climatológicos necesarios para el cálculo de previsión de riego actual y mediante los datos recogidos, efectúa el cálculo de previsión de riego y presenta al usuario la previsión de tiempo de riego necesario

  - **Obtener Riego Semanal:** Este Caso accede al Sistema Externo **"**WoldWeather" para obtener los datos climatológicos necesarios para el cálculo de previsión de riego para siete días y mediante los datos recogidos, efectúa el cálculo de previsión de riego y presenta al usuario la previsión de tiempo de riego necesario

Requisitos Funcionales que cumple el Caso de Uso **SMVisitantes**:

- RF-UC002.
- RF-UC003.
- RF-UC004.
- RF-UC005.

- **Login:** Este Caso de Usa se encarga de proporcionar el formulario de entrada al sistema como Usuario Registrado y por tanto de direccionarlo a la zona Premium del Portal. Incluye los siguientes Casos de Uso:

  - **Introducir Datos Usuario:** Posibilita la introducción de la dirección de Correo Electrónico proporcionada por el Usuario Registrado así como la Password de entrada.

  - **Codificar Datos:** Se codifica la Password del Usuario.

- **Recuperar y Validar Datos:** Este Caso de Uso se encarga de acceder a los datos guardados en Base de Datos y validar que los datos introducidos pertenecen a un Usuario Registrado.

Requisitos Funcionales que cumple:

- RF-UC001.

- **SMPremium:** Este Caso de Uso especifica como un Usuario Registrado puede obtener la previsión de riego del día actual o semanal de acuerdo a su Ciudad o Pueblo y al equipo de riego del que disponga. Extiende los siguientes Casos de Uso:

  - **Guardar Configuración:** Este Caso de Uso guarda en el "Front-End" (lado cliente) del usuario los datos de la Zona de Riego solicitada (ciudad o pueblo y Comunidad Autónoma) y los datos del equipo de riego utilizado, con el fin de que no tenga que introducir dichos datos cuando quiera saber las previsiones de tiempo de riego.

  - **Acceso con Configuración Guardada:** Este Caso de Uso se encarga de obtener la previsión de riego del día actual o semanal de acuerdo a la configuración del usuario guardada mediante el Caso de Uso "Guardar Configuración". Extiende los siguientes Casos de Uso: **Obtener Riego Hoy y Obtener Riego Semanal**, ya descritos anteriormente, por ser comunes al Caso de Uso **SMVisitantes.**

  - **Acceso Nueva Búsqueda:** ya descrito anteriormente, por ser común al Caso de Uso **SMVisitantes.**

  Requisitos Funcionales que cumple el Caso de Uso **SMPremium**:

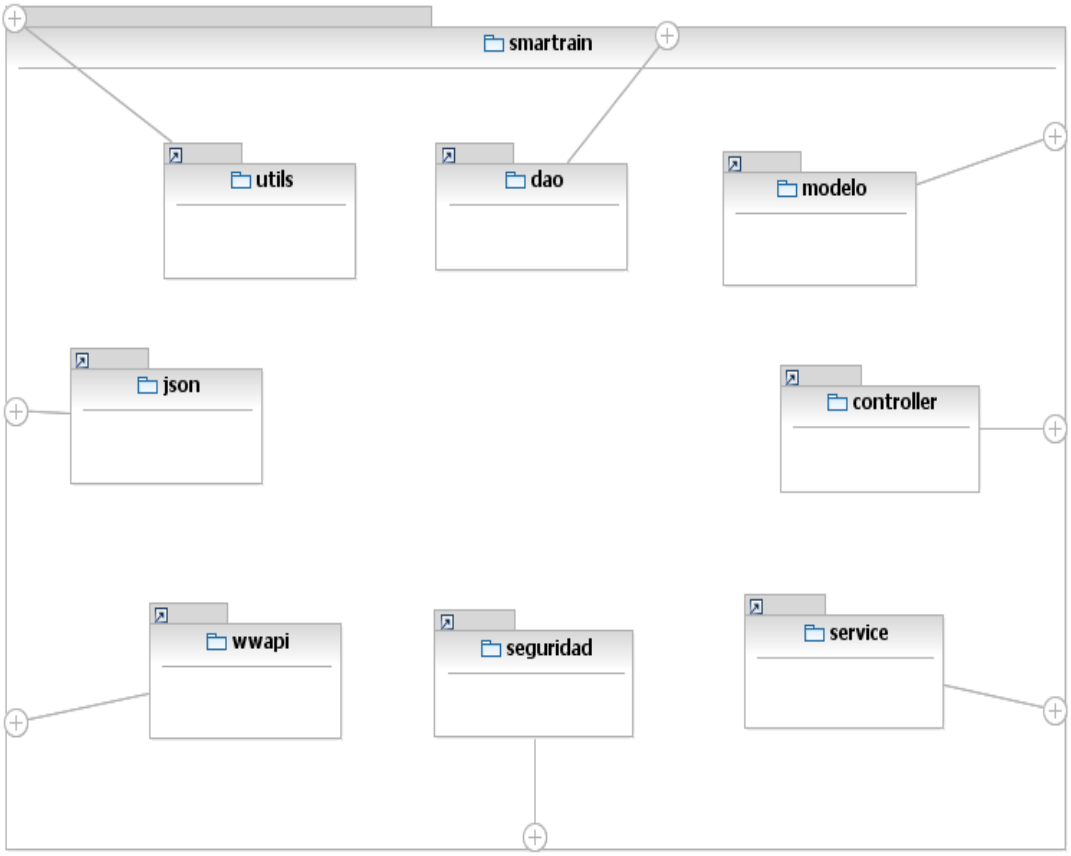
  - RF-UC002.
  - RF-UC003.
  - RF-UC004.
  - RF-UC005.
  - RF-UR002.
  - RF-UR003.

CAPITULO 5. DISEÑO DEL SISTEMA

MODELO DE CLASES DE DISEÑO

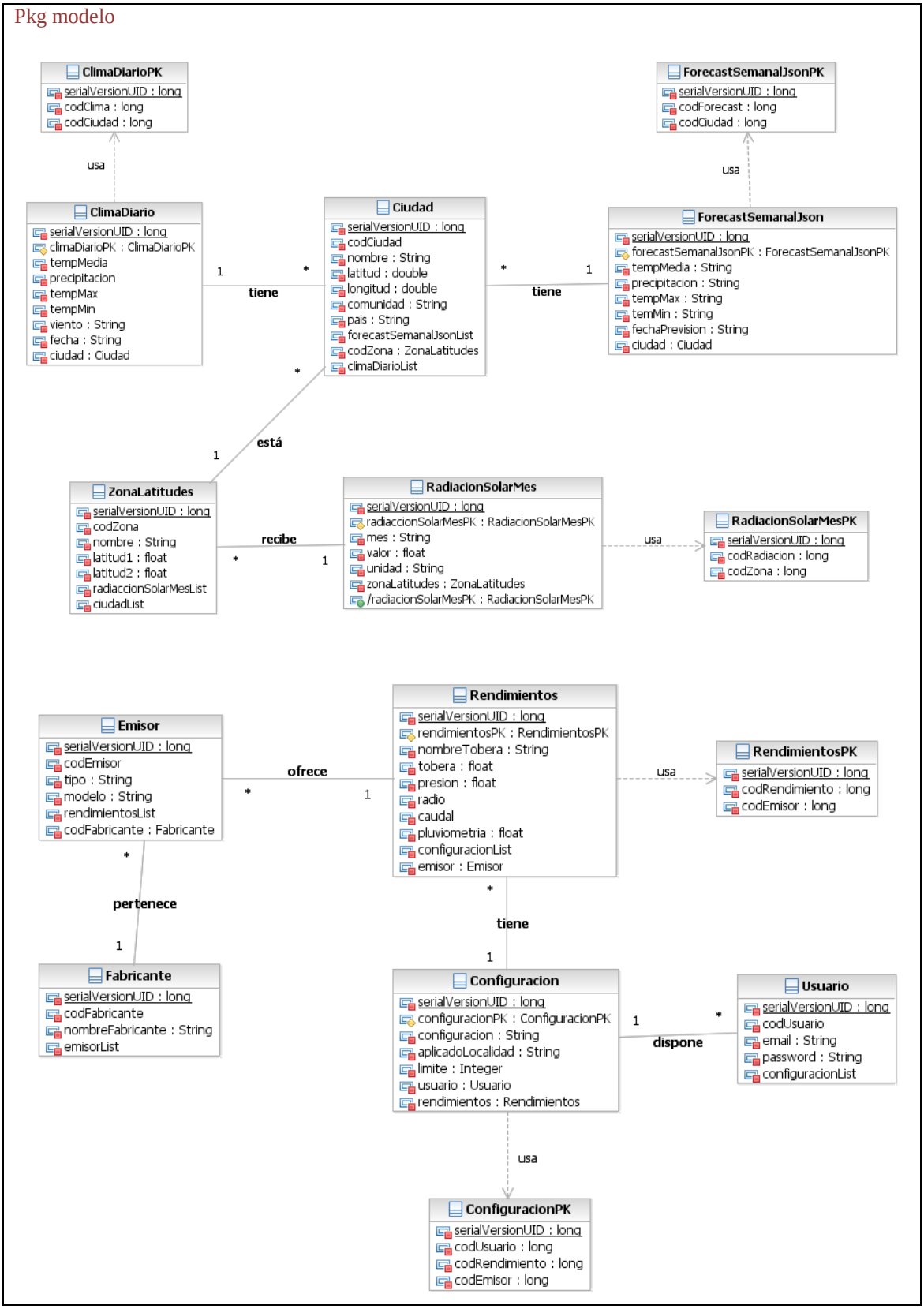
Un diagrama de clases de diseño recoge tanto los conceptos del dominio del sistema propios del diagrama de clases de análisis como aquellos conceptos que forman parte de la implementación de la propia aplicación del Sistema. Una clase del modelo de diseño, además de representar un concepto del mundo real en el que actúa el Sistema, también puede representar un concepto de implementación del sistema que se está modelando.

Paquetes del Sistema



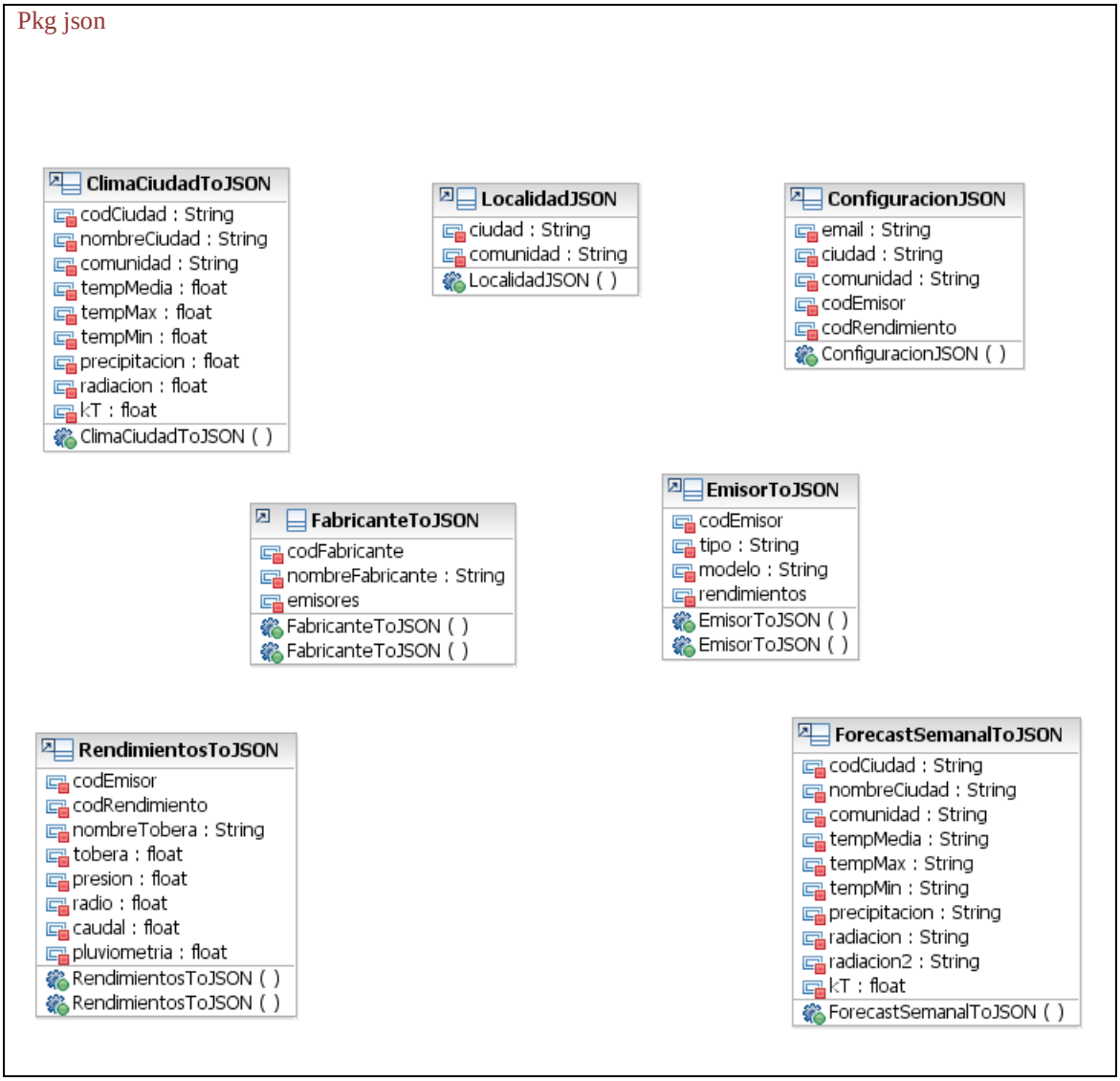
Dominio de Negocio (Modelo de Datos)

Pkg modelo



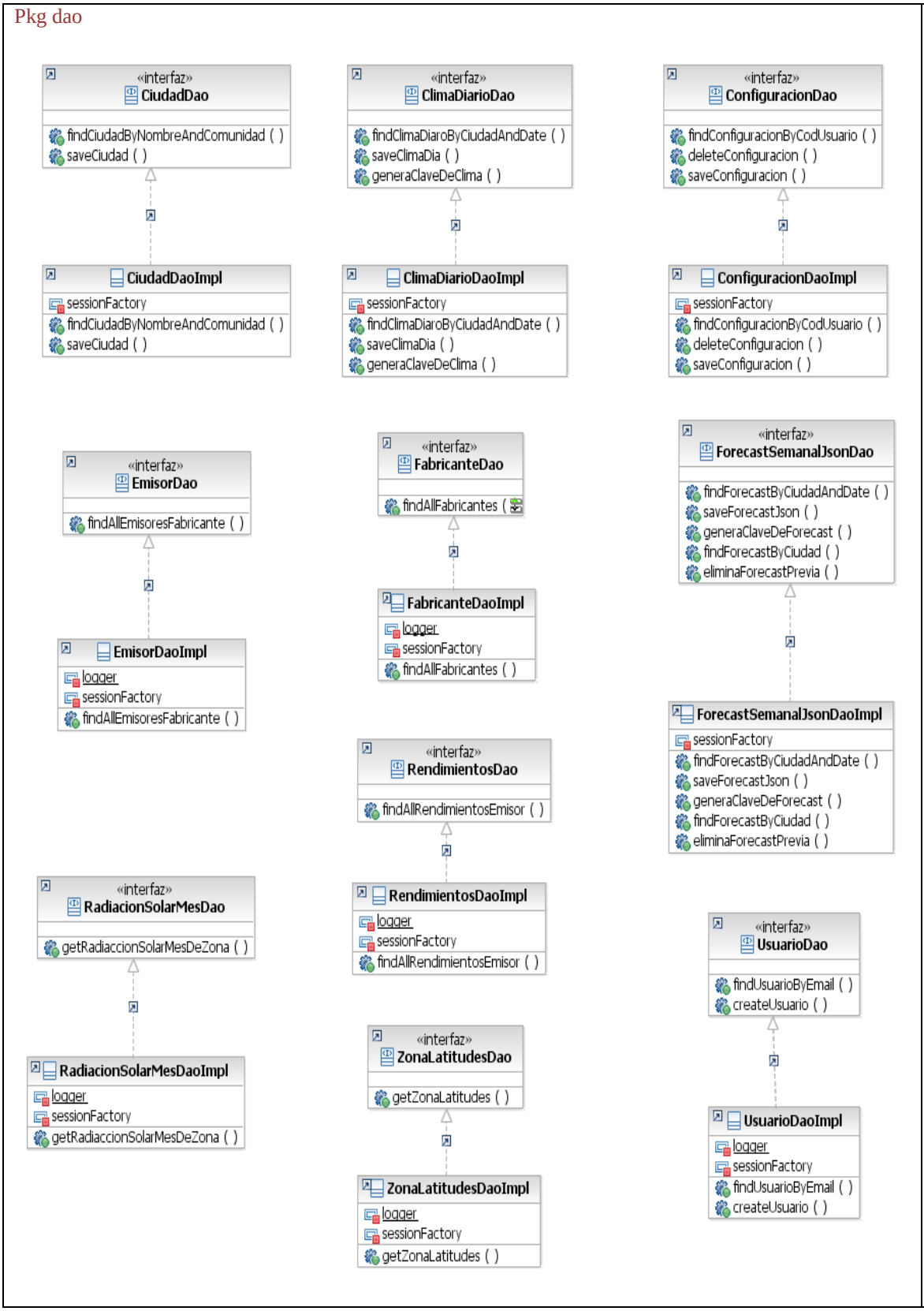
Clases de Datos JSON

Pkg json



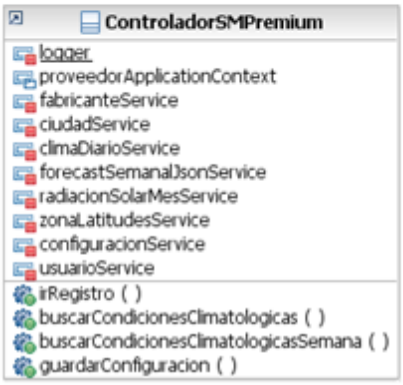
Clases DAO

Pkg dao



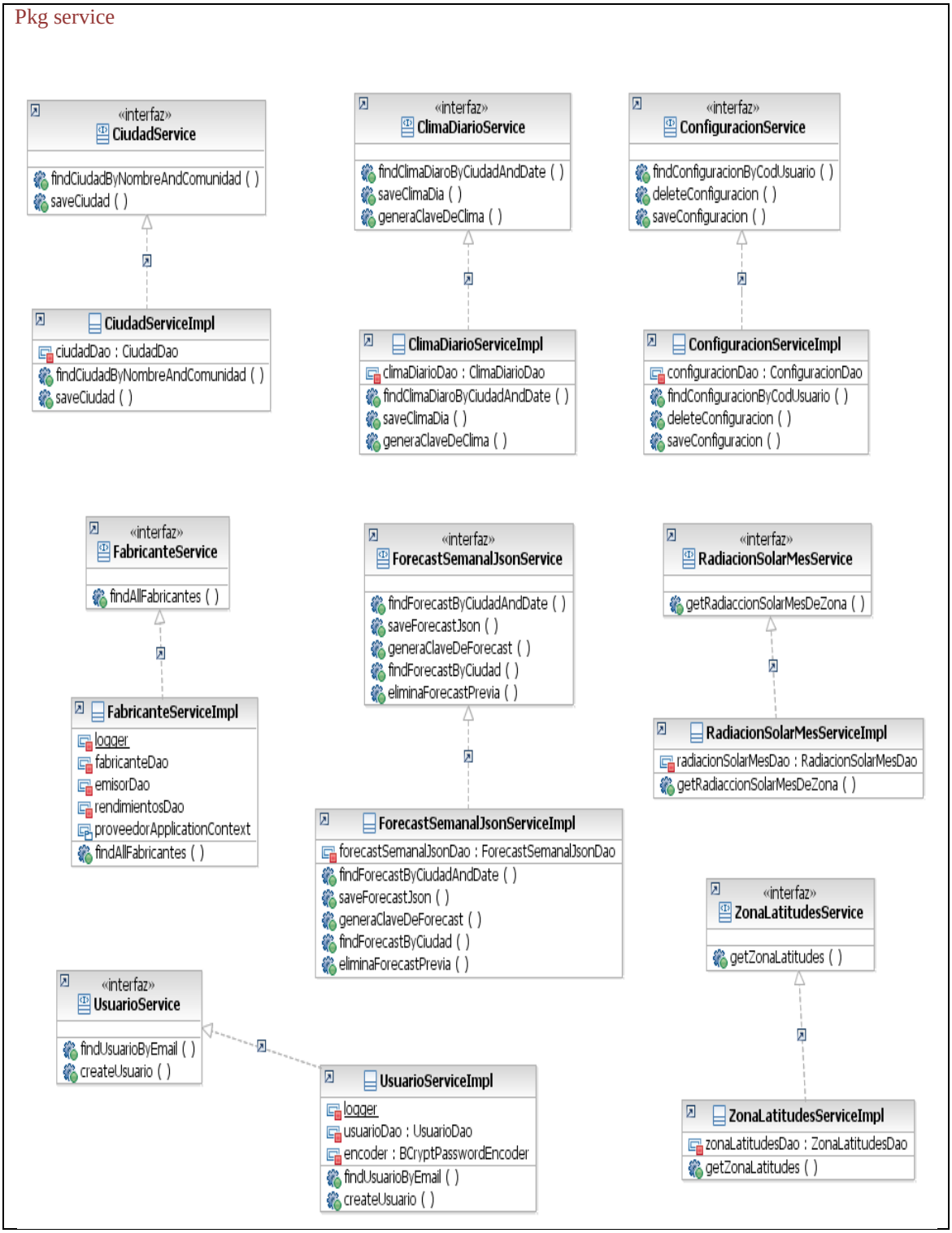
Clases del Controlador

Pkg controller

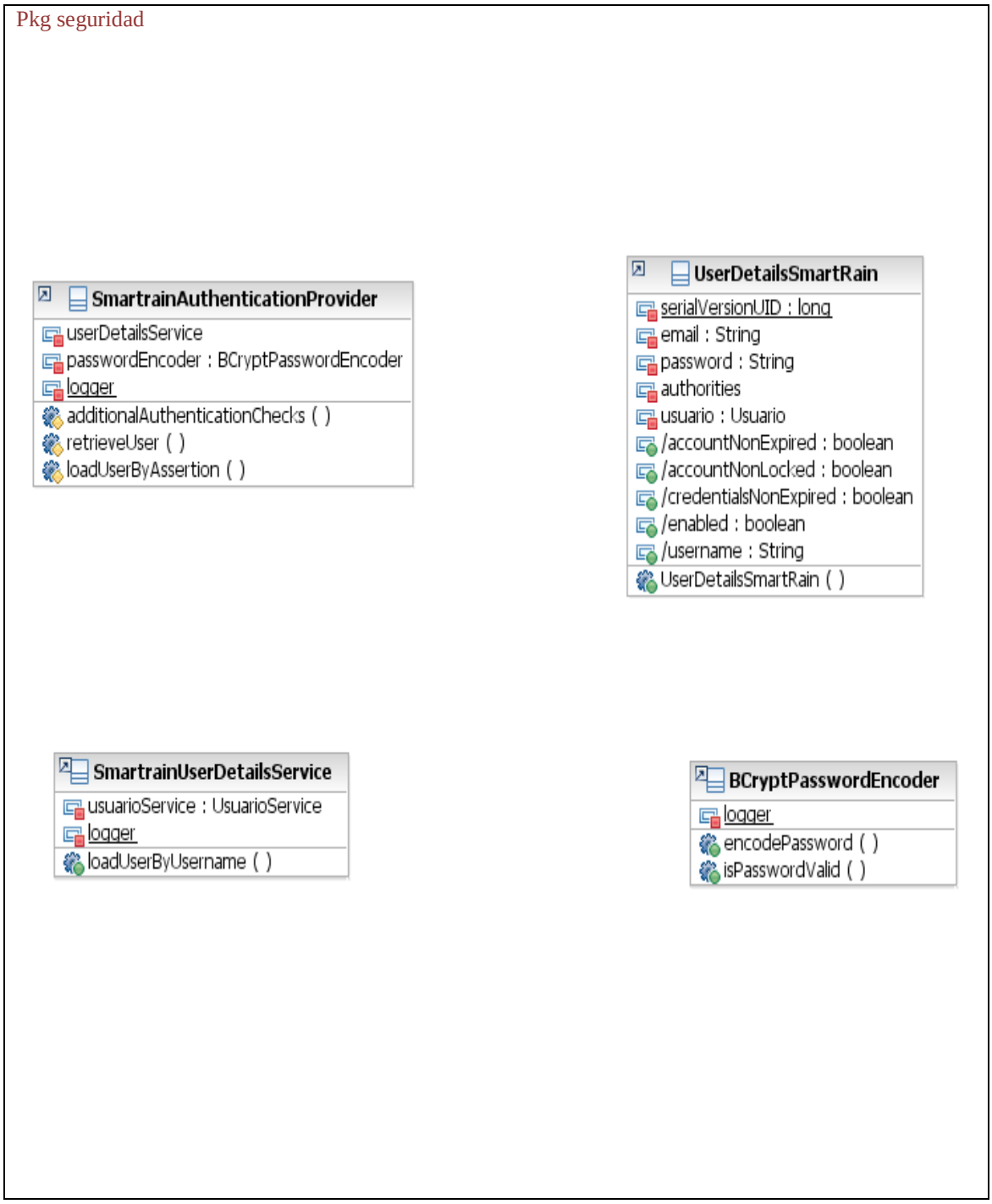


Clases de Servicios

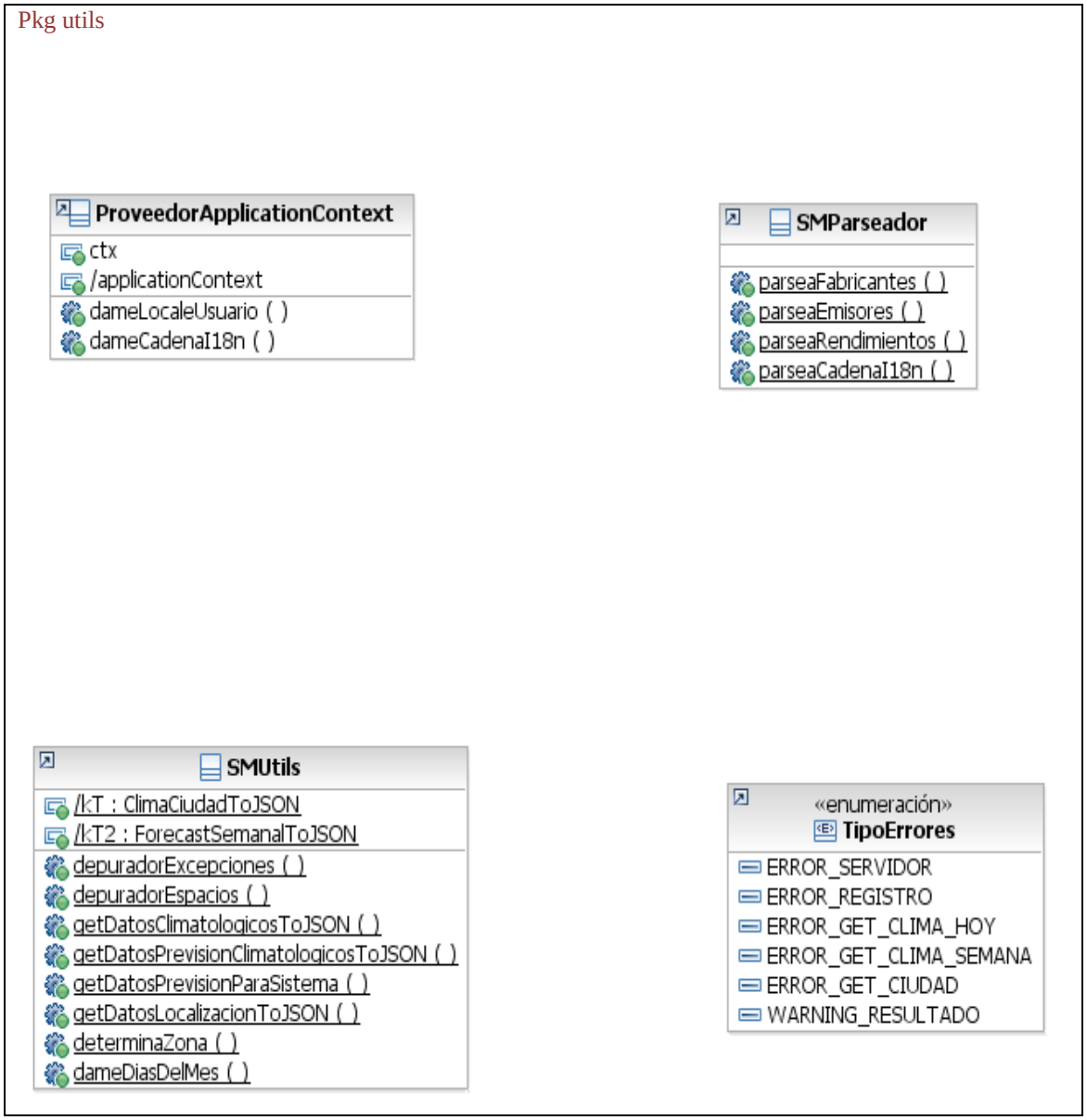
Pkg service



Clases de Seguridad



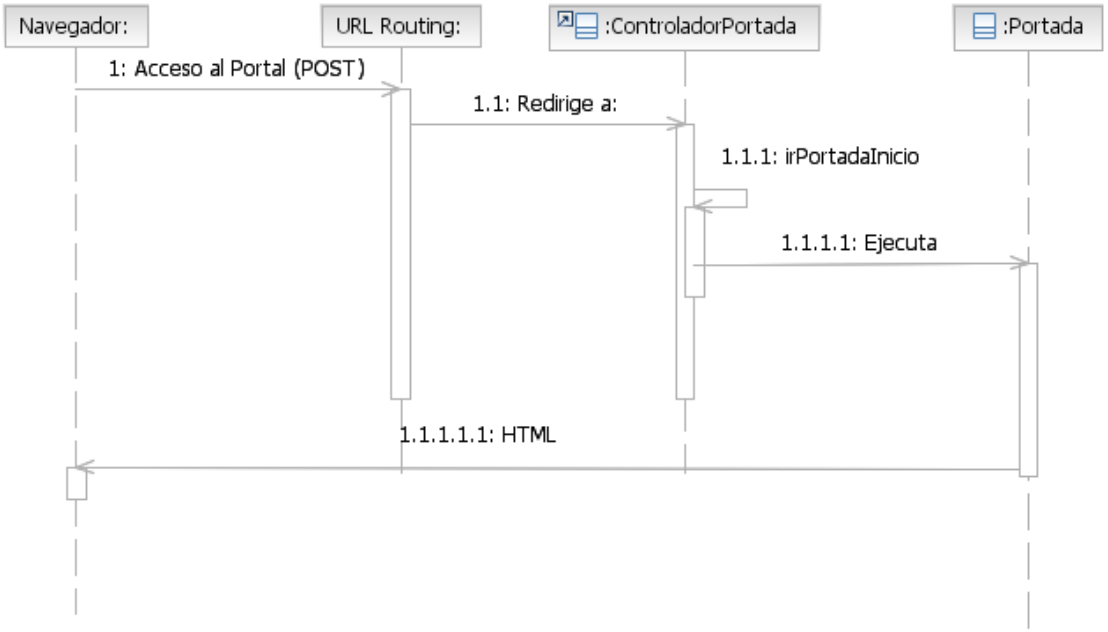
Clases de Utilidad



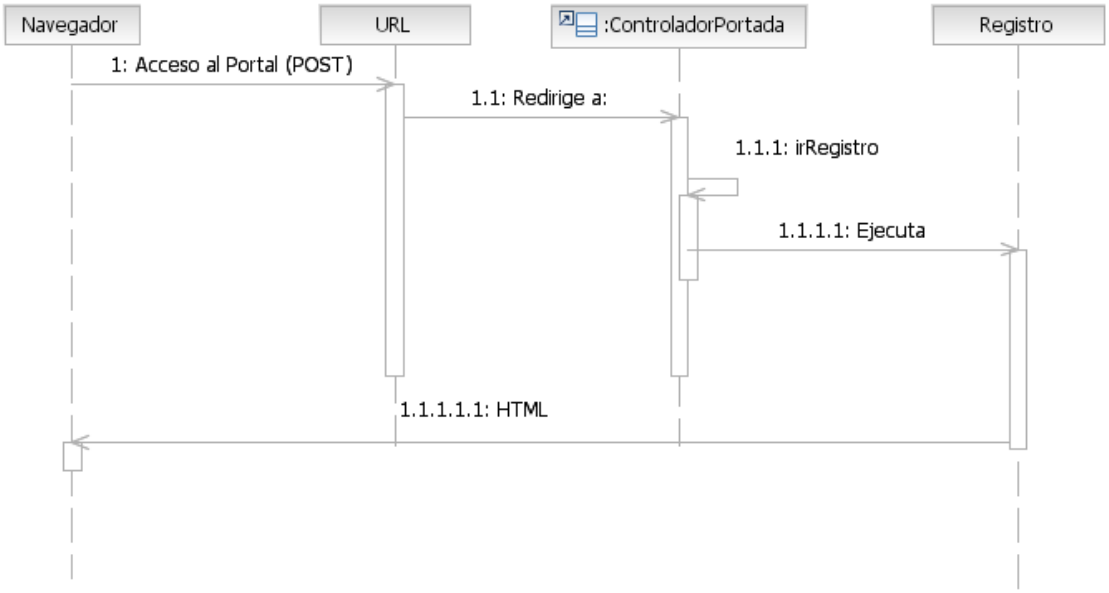
Clases de Acceso a WorldWeather

PRINCIPALES DIAGRAMAS DE SECUENCIA DEL SISTEMA

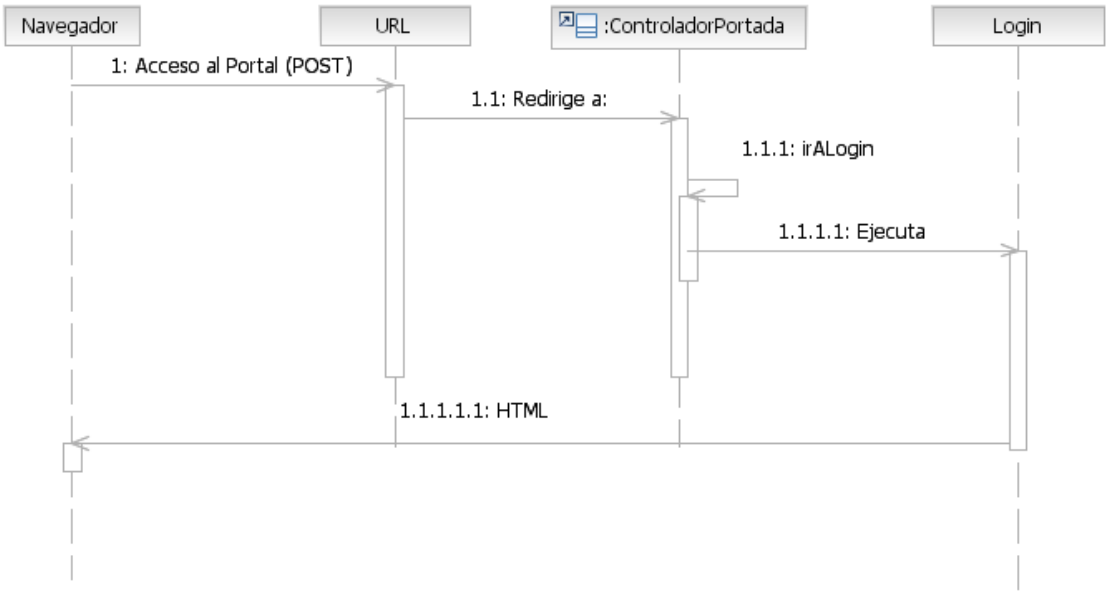
Acceso al Portal Smart Rain



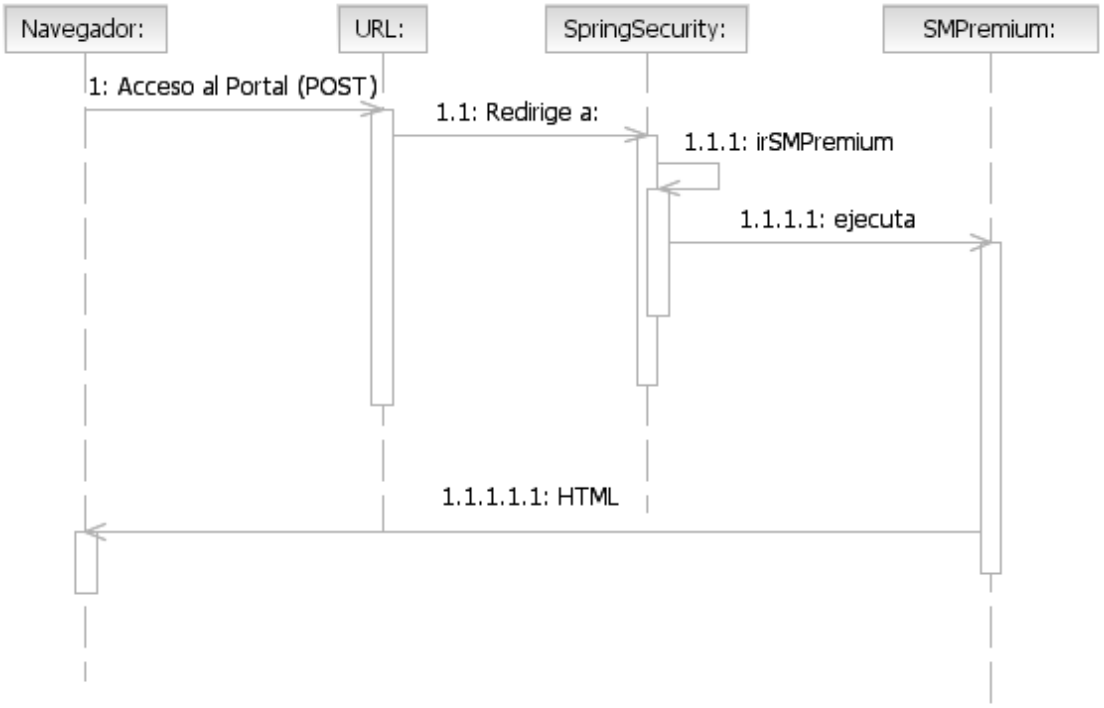
Acceso al Registro



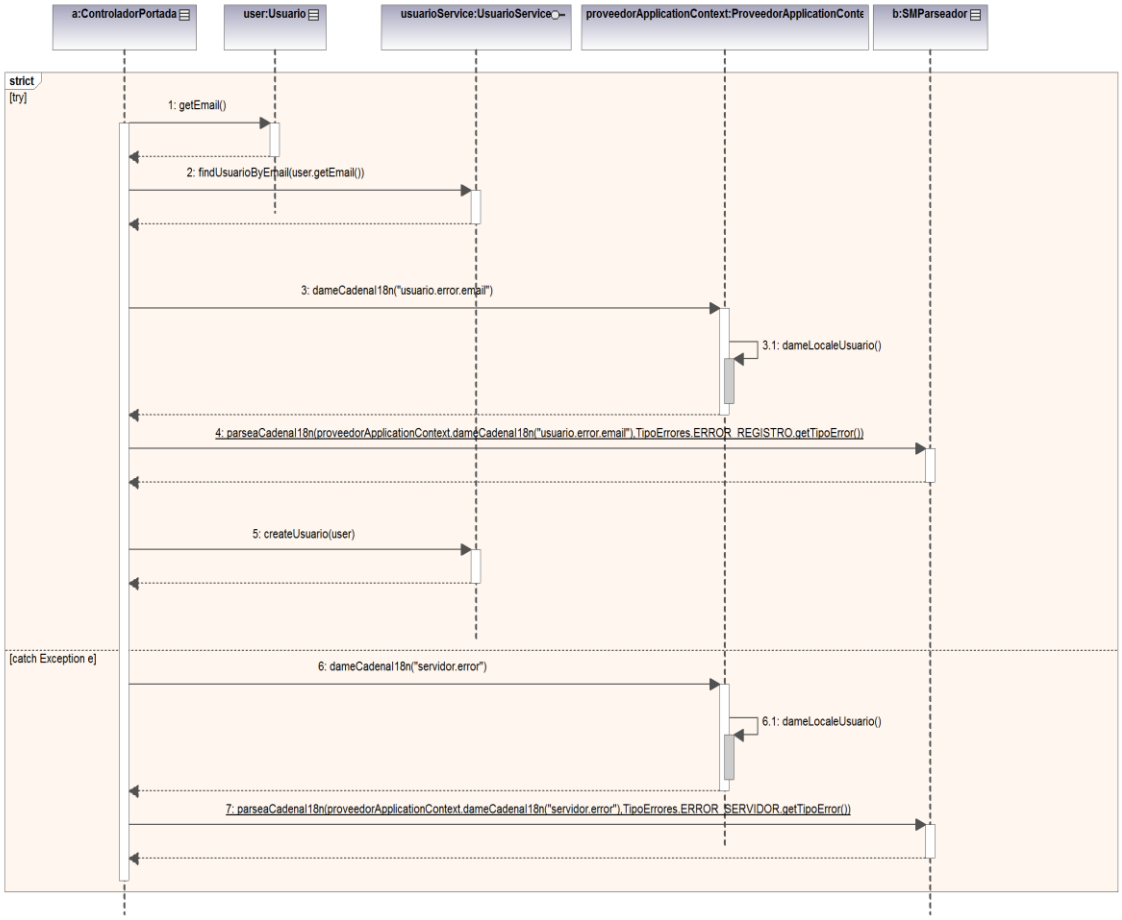
Acceso al Login

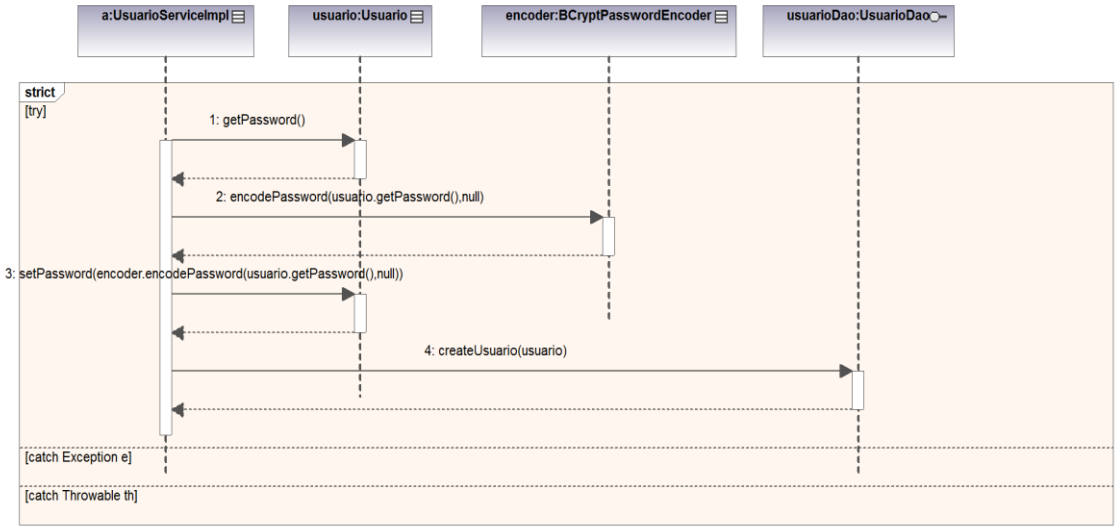


Acceso a Zona Premium

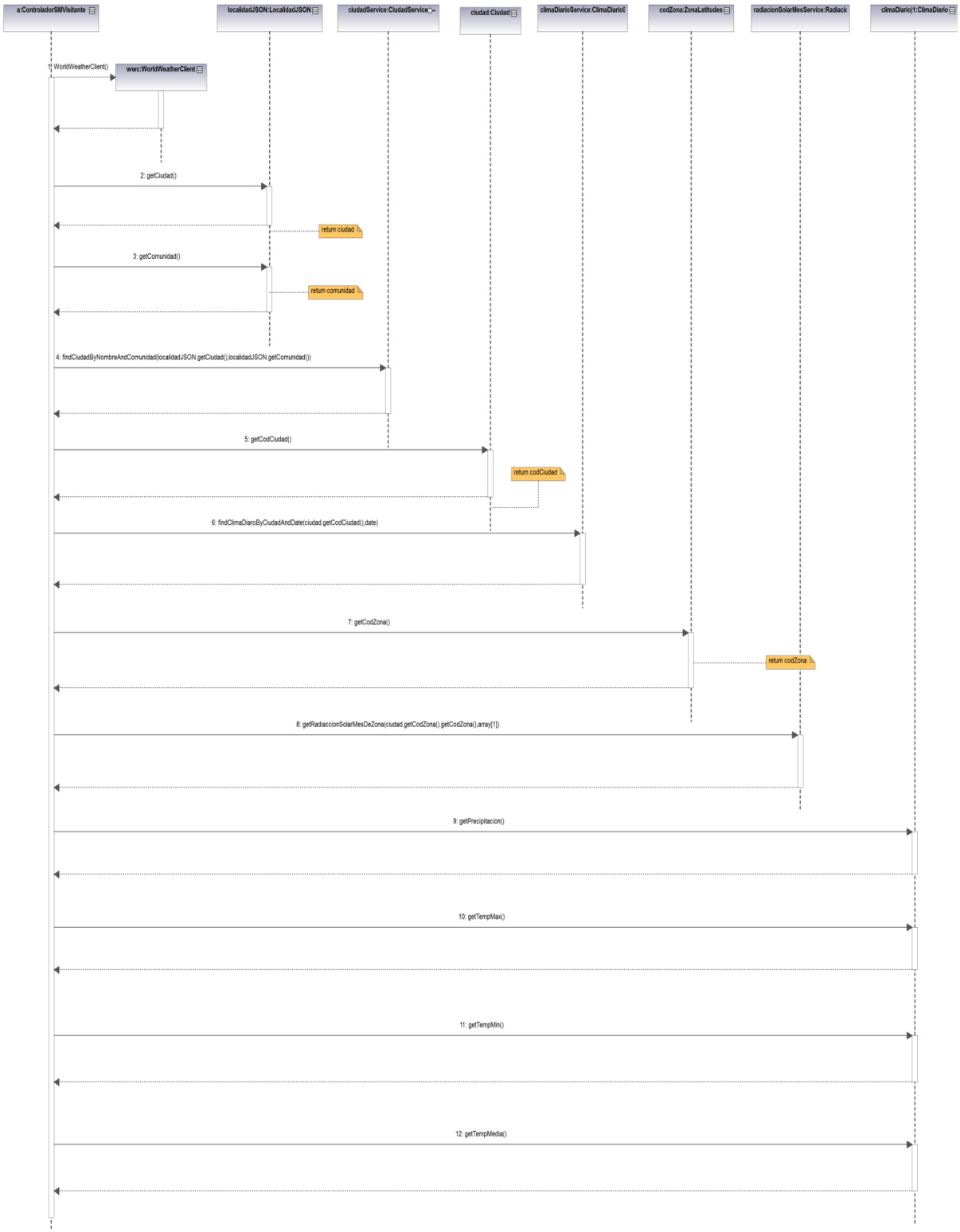


Registrarse

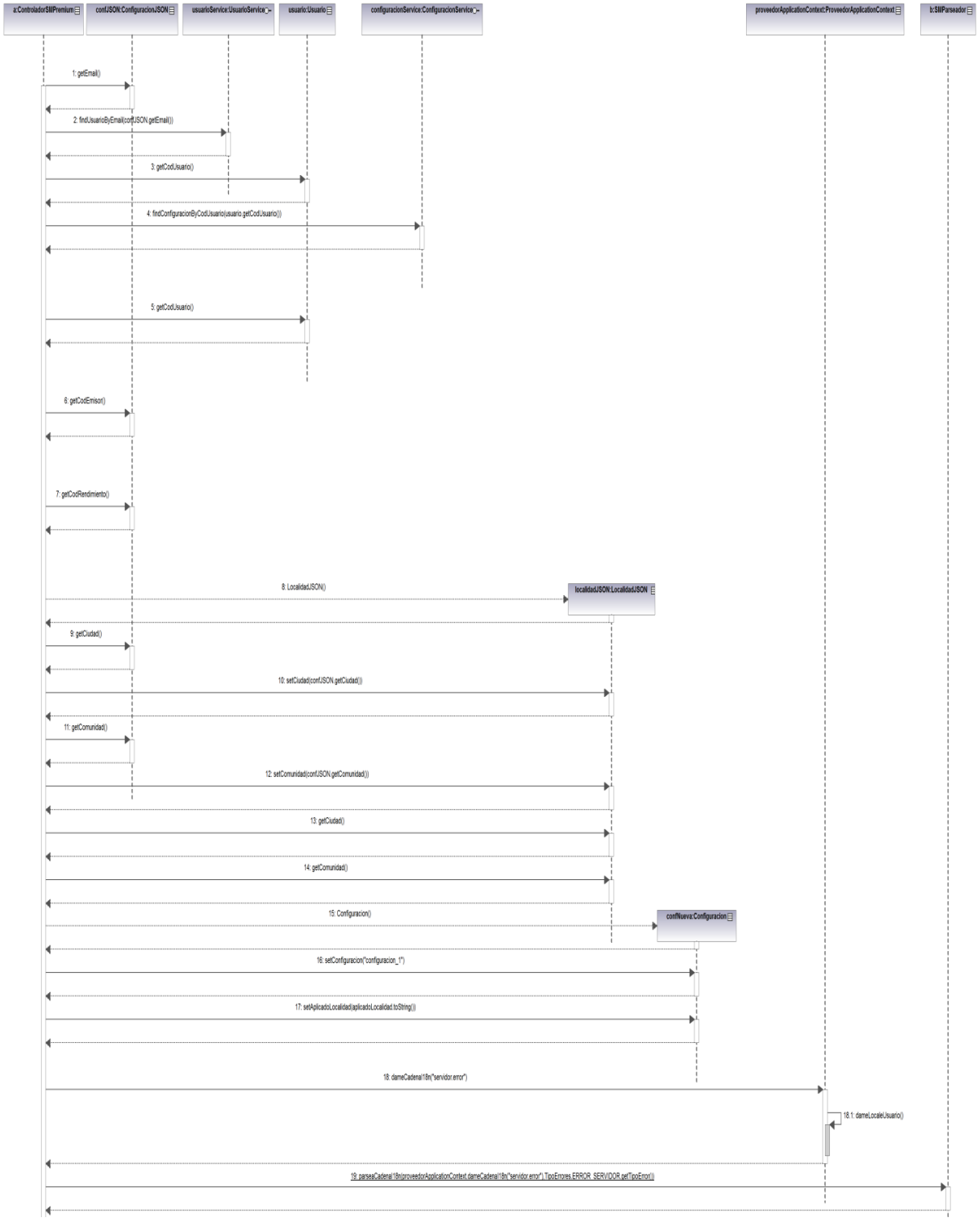




Obtener Tiempo de Riego



Guardar Configuración



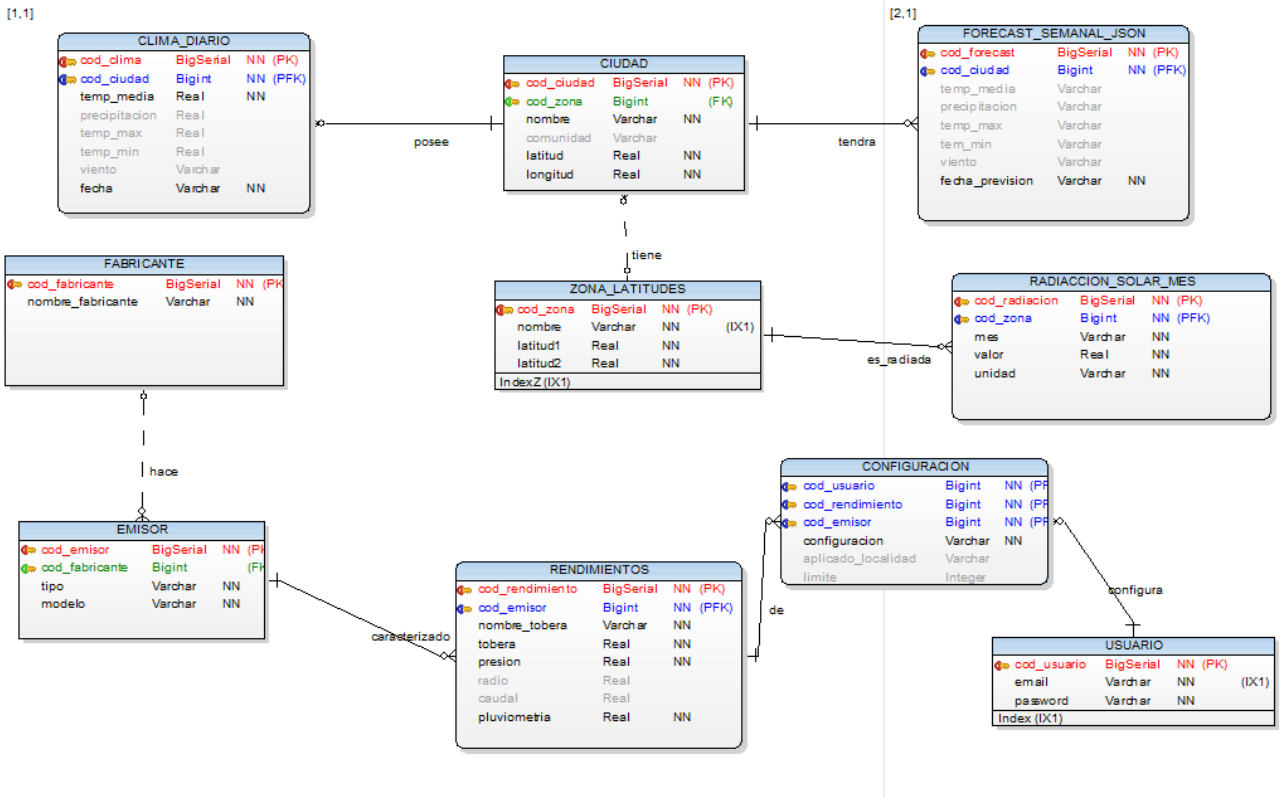
DISEÑO FÍSICO DE DATOS

A partir del diseño físico de datos se pretende representar y definir todos los datos que se introducen, almacenan, transforman y producen dentro del sistema de información. Dado que el modelo de datos es un medio para comunicar el significado de los datos y las relaciones entre ellos, ofrece una serie de ventajas que son las siguientes:

- Comprensión de los datos de una organización y de su funcionamiento.
- Obtención de estructuras de datos independientes del entorno físico.
- Control de los posibles errores desde el principio, o al menos, darse cuenta de las deficiencias lo antes posible.
- Mejora del mantenimiento del Sistema.

Aunque la estructura de datos puede ser cambiante y dinámica, normalmente es mucho más estable que la estructura de procesos. Como resultado, una estructura de datos estable e integrada proporciona datos consistentes que puedan ser fácilmente accesibles según las necesidades de los usuarios, de manera que, aunque se produzcan cambios organizativos, los datos permanecerán estables.

Diagrama relacional



CAPITULO 6. IMPLEMENTACIÓN

Como se ha explicado en el presente documento, el desarrollo del proyecto tiene como objetivo la creación de un sitio web de consulta del tiempo de riego óptimo que es calculado en base a la región donde se quiera regar así como el tipo de emisor de riego que se esté utilizando. A continuación se explicará cómo se ha llevado a cabo y que decisiones se han tomado. Además se podrá observar de manera global como funciona, es decir, como interaccionan cada una de las capas.

DETALLE DE LAS TECNOLOGÍAS

En puntos anteriores se ha descrito el entorno tecnológico utilizado. En esta sección se detallará las versiones aplicadas de cada una de estas tecnologías.

Herramientas

- Servidor y administrador de bases de datos PostgreSql. 9.1.8 (opensource)
- Servidor de aplicaciones WEB JSP: Apache Tomcat 7.0. (opensource)
- Entorno de desarrollo Eclipse Juno (opensource)
- Maven (opensource)
- Hibernate 4 (opensource)
- Entorno de desarrollo NetBeans para replicación en clases de la base de datos mediante JPA (opensource)

Librerías Java

- JDK 1.7 (opensource)
- J2EE 2 (opensource)

Librerías base de datos

- Driver base de datos postgresql9.1-901.jdbc4.jar (opensource)
- Librería Pool base de datos commons-dbcp-1.4.jar (opensource)

Librerías MVC (se detallarán cada una de ellas posteriormente)

- Librerías Spring 3 (opensource)
- Librerías Spring security (opensource)

Scripts Javascript

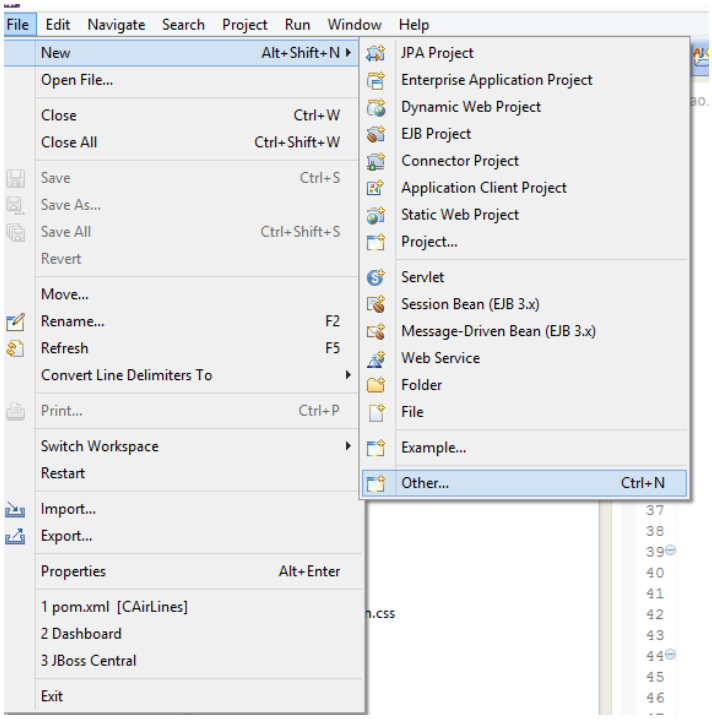
- jquery-2.0.2.js (opensource)
- jquery.validate.min.js (opensource)
- jquery.json-2.4.min.js(opensource)
- bootstrap.min.js(opensource)
- bootstrap.js(opensource)

CSS

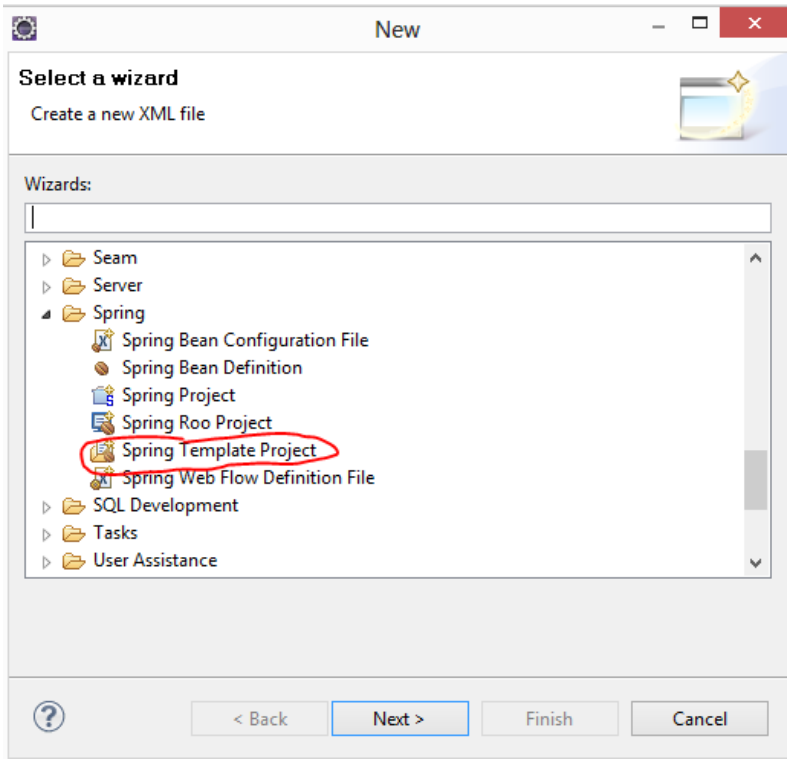
- bootstrap.css

PREPARACIÓN DEL PROYECTO

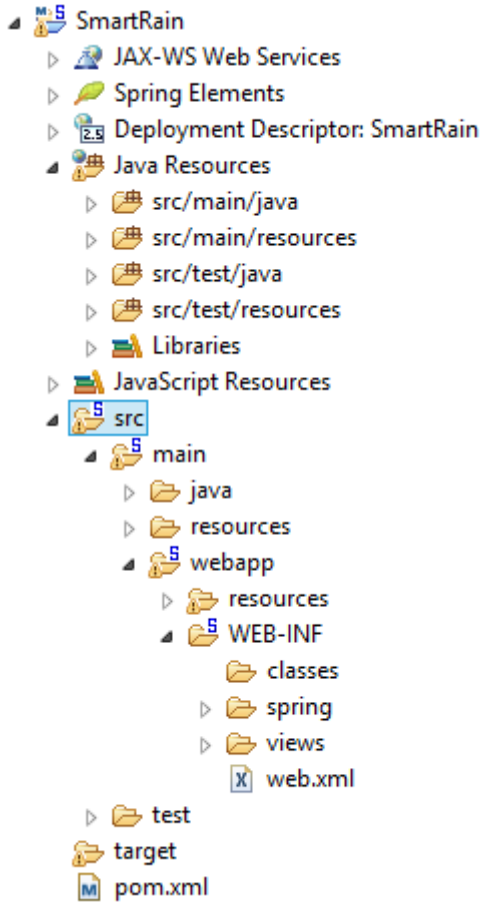
Para crear un proyecto de las características dadas, en primer lugar, en el entorno eclipse seleccionaremos nuevo proyecto-->otro:



Y a continuación pincharemos en Spring Template Project el cual genera la estructura de carpetas para seguir correctamente un patrón MVC además de una integración con maven que permitirá descargar y incluir todas las librerías necesarias de manera automática.



La estructura generada es la siguiente:



Especial atención hay que poner en el archivo pom.xml, éste tiene que ver con maven y ahí se pondrán todas las dependencias del proyecto para así incluir las librerías necesarias. Se incluyen las dependencias de la siguiente manera:

```
<!--****************** SPRING DEPENDENCIES ****************************************** -->

<dependencies>
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-web</artifactId>
                <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-webmvc</artifactId>
                <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-orm</artifactId>
                <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-test</artifactId>
                <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-jdbc</artifactId>
                <version>${org.springframework.version}</version>
        </dependency>
```

```
<!-- Spring Security -->
<dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-core</artifactId>
        <version>${org.spring.security.version}</version>
</dependency>

<dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-web</artifactId>
        <version>${org.spring.security.version}</version>
</dependency>

<dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-config</artifactId>
        <version>${org.spring.security.version}</version>
</dependency>
<dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-taglibs</artifactId>
        <version>${org.spring.security.version}</version>
</dependency>
```

Para más detalle y visualización completa, obsérvese el propio archivo adjuntado a la memoria. Aunque a simple vista pueda parecer complicado, no lo es, ya que en http://mvnrepository.com/ se dispone de un buscador para buscar las librerías y ésta muestra el código a pegar al pom.

El siguiente paso, es cambiar el archivo web.xml en el que hay que configurar spring, indicando que la aplicación se va a regir por el uso de servlets del mismo framework. Además se hará otras referencias a otros archivos xml que habrá que crear para el contexto de hibernate, para la seguridad así como el contexto de los servlets:

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
                /WEB-INF/spring/appServlet/servlet-context.xml
                /WEB-INF/spring/hibernate-context.xml
                /WEB-INF/spring/security.xml
        </param-value>
</context-param>
```

Para más detalle, revisar el archivo en el proyecto adjuntado.

Respecto al archivo servlet-context.xml, lo hay que indicar es donde se encuentran las vistas y los recursos de la aplicación tales como javascripts, css, imágenes, etc.

```
<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
resources in the ${webappRoot}/resources directory -->
<resourcesmapping="/resources/**"location="/resources/"/>

<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
/WEB-INF/views directory -->
<beans:beanclass="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:propertyname="prefix"value="/WEB-INF/views/"/>
        <beans:propertyname="suffix"value=".jsp"/>
</beans:bean>

<context:component-scanbase-package="com.clepsidra.smartrain"/>
```

El archivo de hibernate básicamente se detalla las propiedades de la conexión a la base de datos que se definen en un archivo de texto plano de la siguiente manera:

```
jdbc.driver= org.postgresql.Driver
jdbc.url=jdbc:postgresql://localhost:5432/smartrain
jdbc.user=postgres
jdbc.password=xxxxxxxx
```

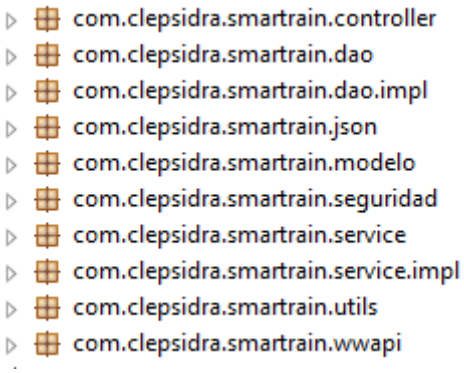
Finalmente en el archivo security.xml se establecen algunos aspectos de seguridad importantes para denegar el acceso a usuarios no registrados a las funcionalidades premium ofrecidas por smart rain. Dichas funcionalidades premium son solo accesibles mediante un sistema de login. El siguiente fragmento de código indica que para acceder a la página premium de smart rain, primero hay que autentificarse, así pues se redirecciona a la página login si se intenta acceder al panel de gestión sin previa autentificación.

```
<intercept-urlpattern="/smpremium*" access="ROLE_USER"/>

<!--            <access-denied-handler error-page=""/> -->

                <form-login
                login-page="/login"
                default-target-url="/smpremium"
                always-use-default-target="true"
                authentication-failure-url="/login?error=-1"/>
```

Para más detalle obsérvese el archivo original.

ESTRUCTURACIÓN DE PAQUETES

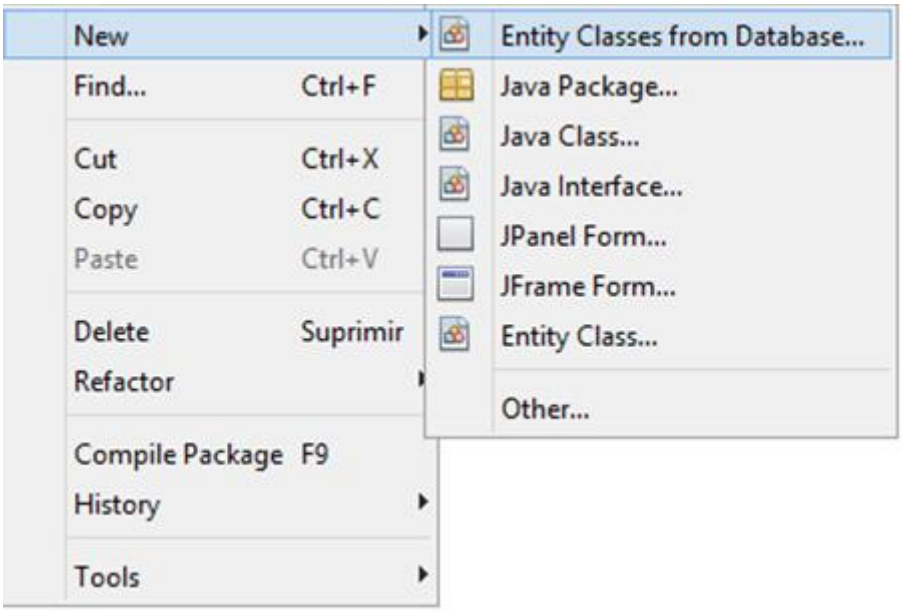
Para seguir el patrón de diseño mvc con spring y usar hibernate tenemos que estructurar las clases e interfaces de una manera ordenada. Para ello se han creado los siguiente paquetes.



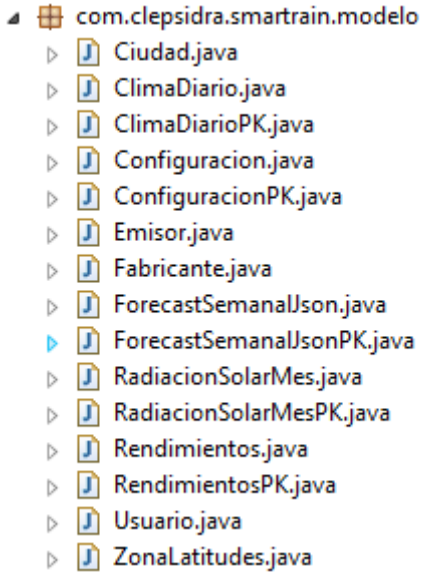
Como se puede observar hay una serie de paquetes relacionados explícitamente con la implementación del patrón de diseño mvc con spring. Tenemos el modelo donde están las clases replicadas de la base de datos. La siguiente capa que es el DAO la cual se comunica con el servicio (capa explícita de spring para aportar mayor seguridad), ambas siguiendo una implementación de interfaces. Y por último encontramos un paquete que contiene los controladores que mapean las acciones realizadas por el cliente. Por otro lado, encontramos otros paquetes extra que aportan funcionalidades adicionales tales como el de seguridad, json, utils y wwapi, este último, de vital importancia, ya que es donde se encuentran las clases necesarias para comunicar el servidor de Smart Rain con el servidor de World Weather y así suministrarse de los datos climatológicos que se requieran.

REPLIACIÓN EN CLASES DE LA BASE DE DATOS, JPA

Para replicar en clases las tablas de la base de datos se ha utilizado Netbeans ya que dispone de una funcionalidad muy útil para dicha replicación. Para ello simplemente en un paquete de un proyecto en Netbeans, hacemos new-->EntityClassesfromDatabase:



Detallando las propiedades de conexión (mismos parámetros que el archivo de texto plano mencionado en puntos anteriores) se generarán las clases del modelo que trasladaremos al proyecto original en Eclipse, tiendo como resultado las siguientes clases:

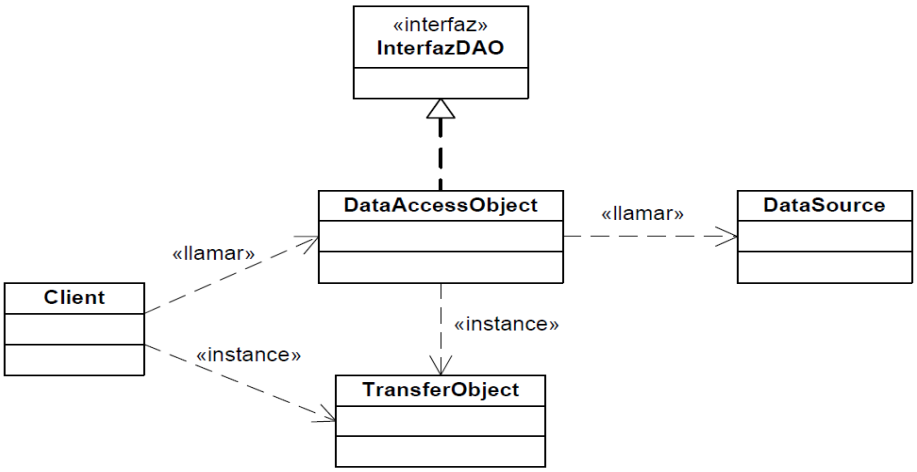


EL PATRÓN DE ACCESO A DATOS

Las operaciones en la base de datos se modelan mediante objetos DAO, Database Access Objects. El patrón DAO encapsula la forma de acceder a la fuente de datos y las operaciones sobre dicha fuente. De esta forma el programador no necesita conocer los detalles del acceso a los datos o de la base de datos a la que accede y sólo debe preocuparse de los datos que necesita.

Como muestra la figura siguiente, se dispone de una interfaz que define los métodos que va a tener el objeto DAO. Habitualmente se definen métodos comunes para la gestión de datos, como listar, borrar, actualizar, etc. Estos métodos suelen identificar las acciones a ejecutar sobre los datos. Los pasos que se realizan son los siguientes:

- El objeto DAO es instanciado por el cliente.
- El cliente solicita los datos al objeto DAO.
- El DAO a su vez solicita los datos a la fuente.
- Por cada fila recibida, el DAO crea un objeto de transferencia, TransferObject,
- que será un bean que tiene como atributos los de la query solicitada.
- Se devuelven los objetos de transferencia al cliente.



Una de las ventajas de utilizar este patrón para modelar el acceso a datos es que habilita la transparencia a los accesos. Los objetos de negocio pueden acceder a las fuentes de datos sin conocer detalles específicos de las mismas. Este acceso es transparente porque los detalles de implementación está encapsulados dentro del objeto DAO.

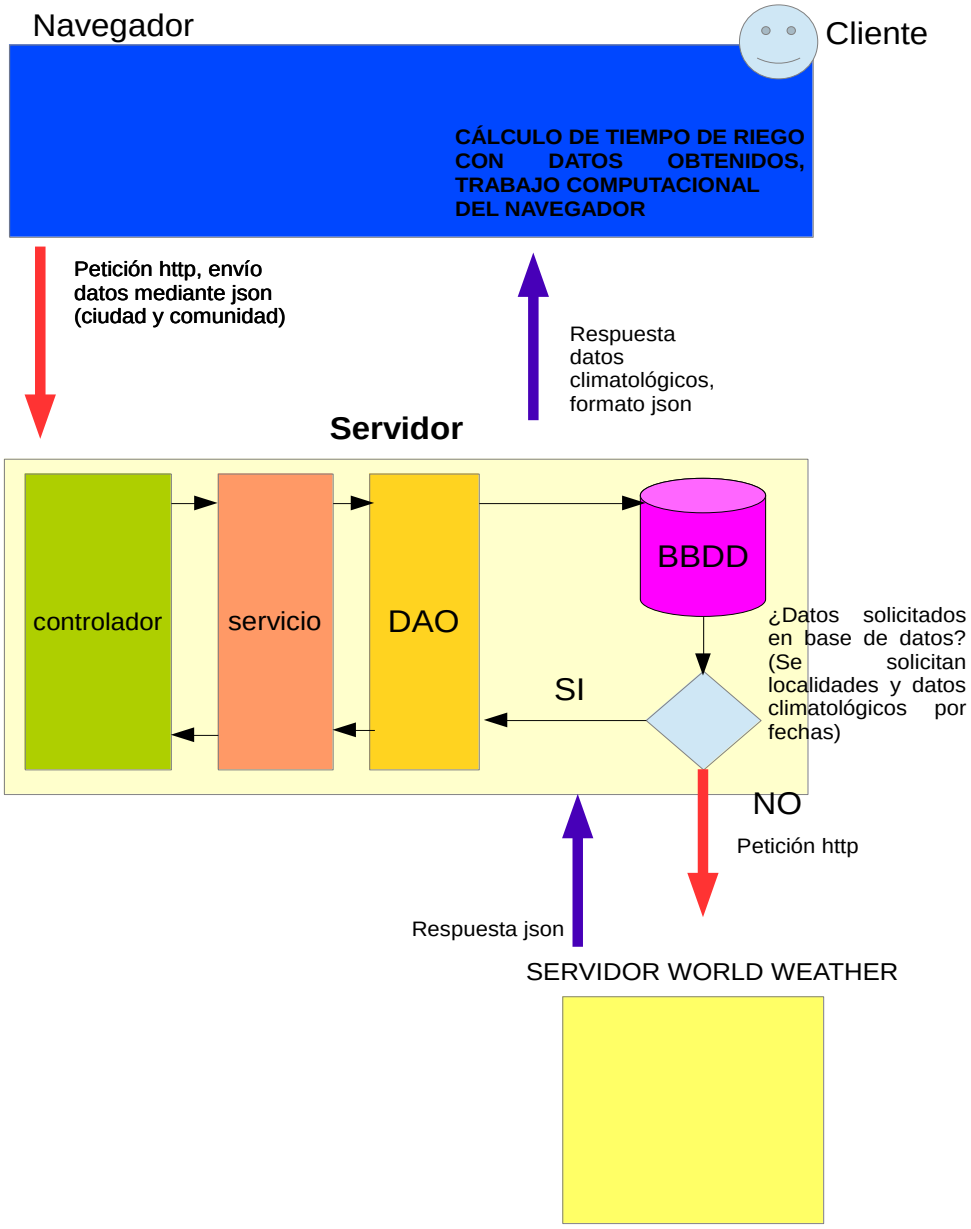
IMPLEMENTACION FUNCIONALIDADES SMART RAIN

Este punto tiene como objetivo mostrar la implementación de alguna de las funcionalidades de la aplicación a modo de ejemplo, de cara a entender todo el desarrollo de la web. Así pues, se verá toda la comunicación entre capas siguiendo el patrón mvc.

Consulta tiempo de riego

Se va a abordar como es procesada la petición del cliente para la obtención del servidor los datos climatológicos necesarios para el cálculo del tiempo de riego óptimo.

Para entender aún mejor la comunicación entre capas, así como el tratamiento de la información se mostrará a continuación un esquema general para la funcionalidad de cálculo de tiempo de riego para el día actual, considérese que el funcionamiento para la previsión semanal funciona de manera semejante.



Proceso

1. El cliente envía una petición de datos climatológicos para el cálculo del tiempo del riego mediante un formulario estándar en HTML5 validado mediante tecnología javascript.

```
<form class="e" id="formBuscar">
    <div class="controls">
        <input class="span12" type="text" name="busBox" id="busBox" placeholder="Escribe tu localidad...">
    </div>
    <div class="control-group">
            <div class="controls">
                <select name="comunidad" id="comunidad" class="span12">
                    <option value="no">Eliga Comunidad Autónoma para centrar la búsqueda</option>
                    <option value="andalucia">Andalucía</option>
                    <option value="aragon">Aragón</option>
                    <option value="asturias">Principado de Asturias</option>
                    <option value="islas baleares">Islas Baleares</option>
                    <option value="pais vasco">País Vasco</option>
                    <option value="canarias">Canarias</option>
                    <option value="cantabria">Cantabria</option>
                    <option value="castilla-la mancha">Castilla La Mancha</option>
                    <option value="castilla y leon">Castilla y León</option>
                    <option value="catalonia">Cataluña</option>
                    <option value="extremadura">Extremadura</option>
                    <option value="galicia">Galicia</option>
                    <option value="madrid">Comunidad de Madrid</option>
                    <option value="murcia">Región de Murcia</option>
                    <option value="navarra">Comunidad Foral de Navarra</option>
                    <option value="la rioja">La Rioja</option>
                    <option value="comunidad valenciana">Comunidad Valenciana</option>
                    <option value="andalucia">Ceuta</option>
                    <option value="andalucia">Melilla</option>
                </select>
            </div>
     </div>
    <p align="justify" id="errorBuscar"></p>
    <button type="submit" class="btn btn-large btn-block btn-primary">Buscar  <i class="icon-search icon-white"></i></button>
</form>
```

Petición:

```
$('#formBuscar').validate({

            submitHandler : function(form) {

                var nombreCiudad = $('#busBox').val();
                nombreCiudad = omitirAcentos(nombreCiudad);

                localidad.ciudad = nombreCiudad;
                localidad.comunidad = $("#comunidad option:selected").val();

                if(menuOp == "hoy"){

                    $.post("buscarCondicionesHoy", {localidad :$.toJSON(localidad)}, function(response) {

                        climaHoyJSON = response;
                        /*Caso devuelve error*/
                        if(climaHoyJSON.tipo != undefined){
                            $('#errorBuscar').text(climaHoyJSON.mensaje);
                            $('#errorBuscar').fadeIn(1200);
                        }else{

                            $('#buscador').fadeOut(200);
                            $('#emisor').fadeIn(1200);
                            $("#barra").attr("style","width: 50%");
                        }

                    },"json"); //fin post
```

2. El controlador procesa la petición y responde al cliente enviado datos en formato JSON, esta es una de las capas más complejas para esta aplicación, hace chequeos de si la información solicitada está en el sistema o es necesario solicitar los datos al servidor de World Weather. Por tanto se pueden dar esencialmente 3 casos:

    - Caso 1: localidad no está en sistema, petición a servidor World Weather

- Caso 2: localidad está en sistema, pero no los datos climatológicos, petición servidor World Weather.

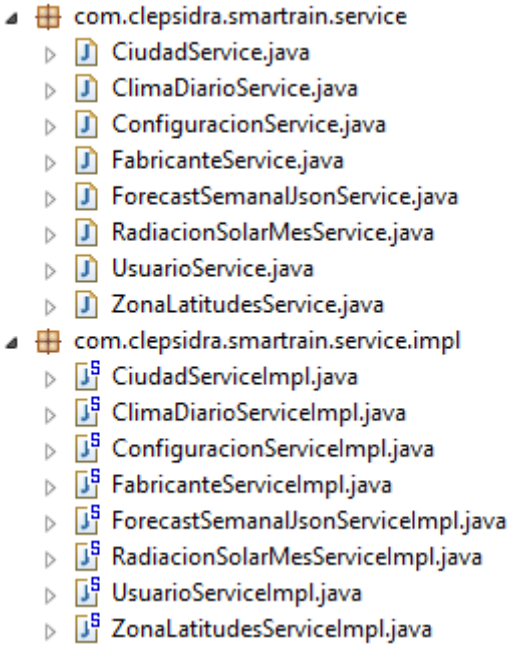
- Caso 3: localidad está en sistema, datos climatológicos también, datos son recuperados del sistema propio.

A continuación se muestra el código del controlador descrito, los casos son fácilmente detectables ya que dicho código tiene los comentarios oportunos.

```
@RequestMapping(value = "/buscarCondicionesHoy", method = RequestMethod.POST)
public @ResponseBody String buscarCondicionesClimatologicas(@RequestParam("localidad") String localidad) throws JsonParseException,
{
    // Para parsear a JSON
    ObjectMapper mapper = new ObjectMapper();

    Ciudad ciudad;
    ClimaDiario climaDiario;
    RadiacionSolarMes rsm;
    LocalidadJSON localidadJSON;
    ClimaCiudadToJSON climaCiudadToJSON;

    SimpleDateFormat formateador = new SimpleDateFormat("yyyy-MM-dd");
    String date;

    /*Solicitador de datos*/
    WorldWeatherClient wwc = new WorldWeatherClient();

    try{
        localidadJSON = mapper.readValue(localidad,LocalidadJSON.class); //Objeto enviado por cliente

        SMUtils.depuradorExcepciones(localidadJSON);

        date = formateador.format(new Date());

        /*Caso de que la ciudad esta en el sistema propio*/
        if(ciudadService.findCiudadByNombreAndComunidad(localidadJSON.getCiudad(), localidadJSON.getComunidad()) != null){

            System.out.println("caso ciudad en sistema");
            ciudad = ciudadService.findCiudadByNombreAndComunidad(localidadJSON.getCiudad(), localidadJSON.getComunidad());

            /*Caso de clima de hoy ya esta en el sistema propio*/
            if(climaDiarioService.findClimaDiaroByCiudadAndDate(ciudad.getCodCiudad(),date) != null){

                System.out.println("clima en sistema");
                climaDiario = climaDiarioService.findClimaDiaroByCiudadAndDate(ciudad.getCodCiudad(),date);
                String array[] = date.split("-");
                rsm = radiacionSolarMesService.getRadiaccionSolarMesDeZona(ciudad.getCodZona().getCodZona(), array[1]);

            /*Caso de que el clima de hoy no está aún en sistema para uan ciudad*/
            }else{

                System.out.println("clima no esta en sistema");
                SMUtils.depuradorEspacios(localidadJSON);

                try{

                    JSONObject respuestaProveedora = wwc.getLocationCityWeather(localidadJSON.getCiudad(),localidadJSON.getComunidad());
                    JSONObject climaHoy = SMUtils.getDatosClimatologicosToJSON(respuestaProveedora);

                    /*Instanciamos clima de hoy*/
                    ClimaDiarioPK claves = new ClimaDiarioPK();
                    claves.setCodCiudad(ciudad.getCodCiudad());
                    claves.setCodClima(climaDiarioService.generaClaveDeClima(ciudad.getCodCiudad()));

                    climaDiario = new ClimaDiario(claves);
                    climaDiario.setTempMin(Float.parseFloat(climaHoy.get("tempMinC").toString()));
                    climaDiario.setTempMax(Float.parseFloat(climaHoy.get("tempMaxC").toString()));
                    climaDiario.setTempMedia((climaDiario.getTempMax() + climaDiario.getTempMin())/2);
                    climaDiario.setPrecipitacion(Float.parseFloat(climaHoy.get("precipMM").toString()));
                    climaDiario.setFecha(date);

                    /*Datos viento*/
                    JSONObject viento = new JSONObject();
                    viento.put("winddir16Point", climaHoy.get("winddir16Point").toString());
                    viento.put("winddirDegree", climaHoy.get("winddirDegree").toString());
                    viento.put("winddirection", climaHoy.get("winddirection").toString());
                    viento.put("windspeedKmph", climaHoy.get("windspeedKmph").toString());

                    climaDiario.setViento(viento.toString());

                    /*Guardamos en sistema el tiempo de hoy para la ciudad buscada"*/
                    climaDiarioService.saveClimaDia(climaDiario);
```

```
                /*Instaciamos la radiacion solar mes*/
                String array[] = date.split("-");
                rsm = radiacionSolarMesService.getRadiaccionSolarMesDeZona(ciudad.getCodZona().getCodZona(), array[1]);

            }catch(JSONException je){

                logger.error("error de parseo debido a respuesta proveedor: "+je.getMessage());
                return SMParseador.parseaCadenaI18n(proveedorApplicationContext.dameCadenaI18n("climaDiario.error.parseo"), TipoErrores.ERROR_GET_CLIMA_HOY.
            }
        }

    /*Caso de que la ciudad no esta en el sistema, peticion*/
    }else{

        System.out.println("ciudad no esta en sistema");
        String nombreCiudad = localidadJSON.getCiudad();
        String nombreComunidad = localidadJSON.getComunidad();

        SMUtils.depuradorEspacios(localidadJSON);

        try{

            JSONObject respuestaProveedora = wwc.getLocationCityWeather(localidadJSON.getCiudad(),localidadJSON.getComunidad());
            JSONObject datosLocalizacion = SMUtils.getDatosLocalizacionToJSON(respuestaProveedora);
            JSONObject climaHoy = SMUtils.getDatosClimatologicosToJSON(respuestaProveedora);

            /*Caso de que coinciden datos de busqueda con respuesta*/
            if(datosLocalizacion.get("comunidad").toString().toLowerCase().equals(nombreComunidad)
                    && datosLocalizacion.get("pais").toString().toLowerCase().equals("spain")){

                ciudad = new Ciudad();
                ciudad.setNombre(nombreCiudad);
                ciudad.setComunidad(datosLocalizacion.get("comunidad").toString().toLowerCase());
                ciudad.setPais(datosLocalizacion.get("pais").toString().toLowerCase());
                ciudad.setLatitud(Double.parseDouble(datosLocalizacion.get("latitud").toString()));
                ciudad.setLongitud(Double.parseDouble(datosLocalizacion.get("longitud").toString()));

                ZonaLatitudes zonaLatitudes = zonaLatitudesService.getZonaLatitudes(SMUtils.determinaZona(ciudad.getLatitud()));
                ciudad.setCodZona(zonaLatitudes);

                ciudadService.saveCiudad(ciudad);

                /* Al ser nueva la ciudad debemos cargarla de la base de datos al guardarla para obtener la PK para guardar clima*/
                Ciudad codCiudad = ciudadService.findCiudadByNombreAndComunidad(ciudad.getNombre(), ciudad.getComunidad());

                /*Instanciamos clima de hoy*/
                ClimaDiarioPK claves = new ClimaDiarioPK();
                claves.setCodCiudad(codCiudad.getCodCiudad());

                climaDiario = new ClimaDiario(claves);
                 climaDiario.setTempMin(Float.parseFloat(climaHoy.get("tempMinC").toString()));
                 climaDiario.setTempMax(Float.parseFloat(climaHoy.get("tempMaxC").toString()));
                 climaDiario.setTempMedia((climaDiario.getTempMax() + climaDiario.getTempMin())/2);
                 climaDiario.setPrecipitacion(Float.parseFloat(climaHoy.get("precipMM").toString()));
                 climaDiario.setFecha(date);

                 /*Datos viento*/
                 JSONObject viento = new JSONObject();
                 viento.put("winddir16Point", climaHoy.get("winddir16Point").toString());
                 viento.put("winddirDegree", climaHoy.get("winddirDegree").toString());
                 viento.put("winddirection", climaHoy.get("winddirection").toString());
                 viento.put("windspeedKmph", climaHoy.get("windspeedKmph").toString());

                 climaDiario.setViento(viento.toString());

                 /*Guardamos en sistema el tiempo de hoy para la ciudad buscada"*/
                 climaDiarioService.saveClimaDia(climaDiario);

                 /*Instaciamos la radiacion solar mes*/
                 String array[] = date.split("-");
                 rsm = radiacionSolarMesService.getRadiaccionSolarMesDeZona(ciudad.getCodZona().getCodZona(), array[1]);

             /*Caso no coinciden datos por inconcluencia de ciudad y comunidad y/o pais*/
             }else{

                 logger.info("Los criterios de busqueda no coinciden con el resultado");
                 return SMParseador.parseaCadenaI18n(proveedorApplicationContext.dameCadenaI18n("crieterio.warning"), TipoErrores.WARNING_RESULTADO.

             }

         }catch(JSONException je){

             logger.error("error de parseo debido a respuesta proveedor/ciudad posiblemente inexistente: "+je.getMessage());
             return SMParseador.parseaCadenaI18n(proveedorApplicationContext.dameCadenaI18n("ciudad.error.parseo"), TipoErrores.ERROR_GET_CIUDAD.get
         }

     }

     /*Construimos respuesta para el cliente*/
     climaCiudadToJSON = new ClimaCiudadToJSON();
     climaCiudadToJSON.setCodCiudad(""+ciudad.getCodCiudad()+"");
     climaCiudadToJSON.setNombreCiudad(ciudad.getNombre());
     climaCiudadToJSON.setComunidad(ciudad.getComunidad());
     climaCiudadToJSON.setPrecipitacion(climaDiario.getPrecipitacion());
     climaCiudadToJSON.setTempMax(climaDiario.getTempMax());
     climaCiudadToJSON.setTempMin(climaDiario.getTempMin());
     climaCiudadToJSON.setTempMedia(climaDiario.getTempMedia());
     climaCiudadToJSON.setRadiacion(rsm.getValor());
     SMUtils.setKT(climaCiudadToJSON);

     return mapper.writeValueAsString(climaCiudadToJSON);

 }catch (Exception e){

     logger.error("error: "+e.getMessage());
     return SMParseador.parseaCadenaI18n(proveedorApplicationContext.dameCadenaI18n("servidor.error"), TipoErrores.ERROR_SERVIDOR.getTipoError()).toString();
 }

}
```

3. Servicio, el método del controlador descrito hace uso de varios servicios para tratar los datos de varias entidades del modelo.

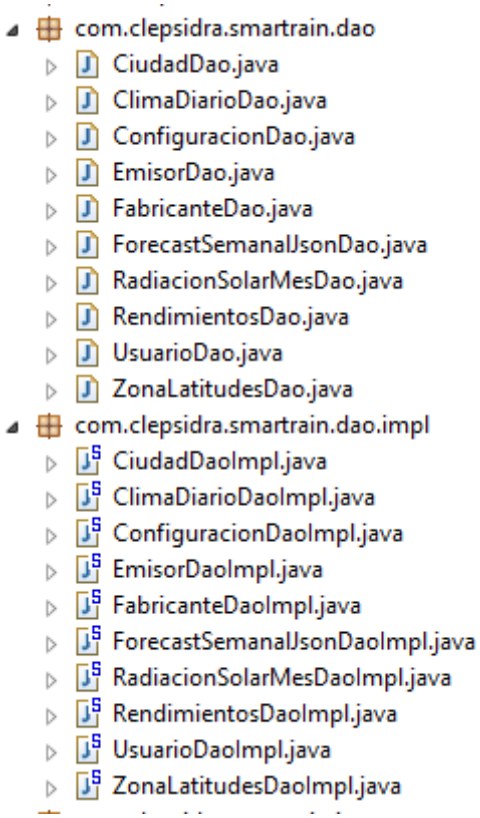


Como se puede observar, la metodología consiste en declararse interfaces por cada servicio en relación al modelo de datos. Por otro lado, en el paquete *com.clepsidra.smartrain.service.impl* se implementan las interfaces mencionadas.

A modo ejemplo, para visualizar el funcionamiento de un servicio, se muestra a continuación el código de los métodos implementados del servicio para la clase ciudad.

```
package com.clepsidra.smartrain.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

@Service
@Transactional
public class CiudadServiceImpl implements CiudadService{

    //~ Instance fields ========================================================================================

    @Autowired
    private CiudadDao ciudadDao;

    @Override
    @Transactional
    public Ciudad findCiudadByNombreAndComunidad(String nombre, String comunidad)throws Exception, Throwable {

        return ciudadDao.findCiudadByNombreAndComunidad(nombre, comunidad);
    }

    @Override
    @Transactional
    public void saveCiudad(Ciudad ciudad) throws Exception, Throwable {

        ciudadDao.saveCiudad(ciudad);
    }

}
```

4. La capa DAO, como se ha podido ver con anterioridad, es usada por el servicio y esta es la última y la responsable de la comunicación con la base de datos, por tanto usa directamente las clases del modelo que analizaremos posteriormente. Al igual que la capa servicio, se establece una serie de interfaces, y clases que implementan dichas interfaces.



Obsérvese el código de la implementación del DAO para la clase ciudad.

```
package com.clepsidra.smartrain.dao.impl;

import org.hibernate.SessionFactory;

@Repository
public class CiudadDaoImpl implements CiudadDao {

    //~ Instance fields ========================================================================================================

    //    private static final Logger logger = Logger.getLogger(ClimaDiarioDaoImpl.class);

    @Autowired
    private SessionFactory sessionFactory;

    @Override
    public Ciudad findCiudadByNombreAndComunidad(String nombre, String comunidad) throws Exception, Throwable {

        Ciudad ciudad = (Ciudad) sessionFactory.getCurrentSession().getNamedQuery("Ciudad.findByNombreAndComunidad")
                .setParameter("nombre", nombre)
                .setParameter("comunidad", comunidad)
                .uniqueResult();
        return ciudad;
    }

    @Override
    public void saveCiudad(Ciudad ciudad) throws Exception,Throwable{

        sessionFactory.getCurrentSession().save(ciudad);
    }

}
```

En esta capa se realizan finalmente las consultas y/o operaciones requeridas sobre la base de datos mediante el objeto sessionFactory, resultado de utilizar Hibernate como motor de persistencia.

Por otro lado, destacar la importancia de las clases del modelo, para el este ejemplo concreto, se está utilizando el DAO de ciudad, por tanto véase a continuación el código del modelo para la clase específica de ciudad (Importante, uso de JPA).

```
 * @author Ivan
 */
@Entity
@Table(name = "\"CIUDAD\"")
@XmlRootElement
@NamedQueries({
//    @NamedQuery(name = "Ciudad.findAll", query = "SELECT c FROM Ciudad c"),
//    @NamedQuery(name = "Ciudad.findByCodCiudad", query = "SELECT c FROM Ciudad c WHERE c.codCiudad = :codCiudad"),
//    @NamedQuery(name = "Ciudad.findByNombre", query = "SELECT c FROM Ciudad c WHERE c.nombre = :nombre"),
//    @NamedQuery(name = "Ciudad.findByLatitud", query = "SELECT c FROM Ciudad c WHERE c.latitud = :latitud"),
//    @NamedQuery(name = "Ciudad.findByLongitud", query = "SELECT c FROM Ciudad c WHERE c.longitud = :longitud"),
    @NamedQuery(name = "Ciudad.findByNombreAndComunidad", query = "SELECT c FROM Ciudad c WHERE c.nombre = :nombre AND c.comunidad = :comunidad")
})
public class Ciudad implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO, generator = "\"CIUDAD_cod_ciudad_seq\"")
    @SequenceGenerator(name = "\"CIUDAD_cod_ciudad_seq\"", sequenceName = "\"CIUDAD_cod_ciudad_seq\"")
    @Basic(optional = false)
    @Column(name = "cod_ciudad")
    private Long codCiudad;
    @Basic(optional = false)
    @Column(name = "nombre")
    private String nombre;
    @Basic(optional = false)
    @Column(name = "latitud")
    private double latitud;
    @Basic(optional = false)
    @Column(name = "longitud")
    private double longitud;
    @Column(name = "comunidad")
```

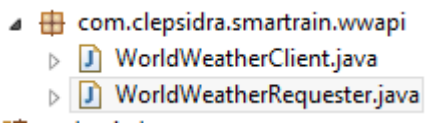
En rojo, se señala una consulta predefinida que se está utilizando en el DAO de Ciudad tal y como se puede ver en el código mostrado con anterioridad.

Peticiones a otro servidor

Esta es una de las funcionalidades que tienen más importancia ya que para el funcionamiento de la aplicación propuesta es necesario suministrarse de los datos climatológicos.

Por tanto, y también desde el punto de vista técnico es interesante también mostrar la implementación de una conexión/petición que se realiza de un servidor a otro, normalmente atípica en aplicaciones corrientes que siguen una arquitectura C/S.

Para que Smart Rain se nutra de los datos climatológicos requeridos para su funcionamiento necesita hacer peticiones, en este caso, al servidor de World Weather, que nos proporciona los datos solicitados en formato JSON. Para ello se ha desarrollado las siguientes clases:



La Clase *WorldWeatherRequester* es la encargada de abrir y cerrar conexiones con el servidor de World Weather y en última instancia de enviar la solicitud de datos requeridos. Por otro lado, con la Clase *WorldWeatherClient* construimos las solicitudes acordes a las peticiones realizadas y en dicha clase usamos objetos del *Requester.*

WorldWeatherRequester:

```
/**
 *
 * @param request
 * @param protocolo
 * @return String, respuesta proporcionada por el servidor al que se ha realizado la peticion.
 */
public static String conexionGET(String request, String protocolo) {

    String response = "";
    BufferedReader rd = null;

    try {

        URL url = new URL(request);

        if (protocolo.equals("HTTP")) {

            URLConnection connection = (URLConnection) url.openConnection();
            rd = new BufferedReader(new InputStreamReader(connection.getInputStream()));

        } else {

            /* PARA CONEXIONES HTTPS, DE MOMENTO NO NECESARIO */
            return null;
        }

        String line;

        while ((line = rd.readLine()) != null) {

            // Process line...
            response += line;
        }

    } catch (Exception e) {

        logger.error("Fallo en la solicitud, causa: " + e.getMessage());

    } finally {

        if (rd != null) {
            try {
                rd.close();
            } catch (IOException ex) {
                logger.error("Problema al cerrar el objeto lector, causa: "
                        + ex.getMessage());
            }
        }
    }

    return response;
}
```

WorldWeatherClient:

Como se ha adelantado anteriormente, esta clase es la encargada de construir las solicitudes para el servidor del World Weahter. Para ello es necesario declararse la siguiente variables:

```
static private final String API = "[redacted]";

private String baseWorldWeatherUrl = "http://api.worldweatheronline.com/free/v1/weather.ashx?q=";
```

En la variable API se establece la clave proporcionada por World Weather para poder solicitarle datos y por otro lado, la variable *baseWorldWeatherUrl* es la URL base para construir las peticiones.

Por el momento, se han implementado dos métodos para consultar a World Weather el clima de una determina ciudad para el día actual, así como una previsión meteorológica para una semana. Además World Weather proporciona datos de localización, útiles para Smart Rain para el tema de cálculos de la radiación incidente terrestre.

Método Clima actual para una ciudad:

```
/**
 *
 * @param ciudad
 * @param pais
 * @return Objeto Json con información relativa a la posición geográfica de la ciudad buscada
 * @throws Exception
 */
public JSONObject getLocationCityWeather(String ciudad, String comunidad) throws Exception{

    try{

        String request = baseWorldWeatherUrl;
        System.out.println("comunidad: "+comunidad);
        request += ciudad;

        request += "%2C"+comunidad;

        request += ("&format=json&num_of_days=1&cc=no&includelocation=yes&key="+API);

        JSONObject jsonObject = new JSONObject(WorldWeatherRequester.conexionGET(request,"HTTP"));

        return jsonObject;

    }catch(Exception e){

        logger.error("Error al parsear y obtener condiciones, causa: "+e.getMessage());
        JSONObject jsonObjectError = new JSONObject("{\"status\":\"error\",\"causa\":\""+e.getMessage()+"\"}");
        return jsonObjectError;
    }
}
```

Método previsión semanal para una ciudad:

```
/**
 *
 * @param ciudad
 * @param pais
 * @return Objeto json con el clima actual de la ciudad encontrada, así como de la próxima semana
 * @throws Exception
 */
public JSONObject getForecastWeatherAndLocationData(String ciudad, String comunidad) throws Exception{

    try{

        String request = baseWorldWeatherUrl;
        System.out.println("comunidad: "+comunidad);
        request += ciudad;

        request += "%2C"+comunidad;

        request += ("&format=json&num_of_days=7&cc=no&includelocation=yes&key="+API);

        JSONObject jsonObject = new JSONObject(WorldWeatherRequester.conexionGET(request,"HTTP"));

        return jsonObject;

    }catch(Exception e){

        logger.error("Error al parsear y obtener previsiones condiciones, causa: "+e.getMessage());
        JSONObject jsonObjectError = new JSONObject("{\"status\":\"error\",\"causa\":\""+e.getMessage()+"\"}");
        return jsonObjectError;
    }
}
```

Base de datos en local

Cabe destacar, el uso de bases de datos en local para la implementación de la funcionalidad que permite al usuario guardar sus configuraciones de búsqueda. En esencia, se ha realizado un sistema mixto donde se guardan los datos de búsqueda tanto en el servidor como en el **navegador** del cliente, todo ello con vista a la evolución de Smart Rain como una aplicación móvil tal y como se comentará en puntos posteriores.

Para ello, se ha aprovechado una de las nuevas ventajas que ofrece el estándar HTML5 que permite desarrollar en el navegador bases de datos en local en SQLite.

A continuación se mostrarán los métodos en javascript que crean una base de datos simple y en la que se hacen operaciones sencillas de búsqueda, actualización e insertado de datos.

Inicialización de Base de datos:

```
function initDatabase() {
    try {
        if (!window.openDatabase) {
            alert('La bases de datos no son compatibles en este navegador.');
            cargaSelects();
        } else {
            var shortName = 'SMDB';
            var version = '1.0';
            var displayName = 'SmartRain Database';
            var maxSize = 2*1024*1024; //  2 Mega bytes
            SMDB = openDatabase(shortName, version, displayName, maxSize);
            createTables();
            selectConfiguracion();
        }
    } catch(e) {

        if (e == 2) {
            // Version number mismatch.
            console.log("Invalid database version.");
        } else {
            console.log("Unknown error "+e+".");
        }
        return;
    }
}
```

El método mostrado es el primero en ejecutarse respecto al funcionamiento de la base de datos una vez se carga la página en el cliente. A destacar, el chequeo que se realiza acorde a la compatibilidad del navegador con esta novedad de HTML5.

En este mismo método son llamados otros dos desarrollados por el autor del presente documento, el que sigue es *createTables():*

```
function createTables(){
    SMDB.transaction(
        function (transaction) {
            transaction.executeSql('CREATE TABLE IF NOT EXISTS configuracion(id INTEGER NOT NULL PRIMARY KEY,configuracion_json TEXT NOT NULL);', []);
        }
    );
}
```

Y seguidamente se comprueba si hay configuraciones previamente guardadas mediante el método *selectConfiguracion():*

```
function selectConfiguracion(){
    SMDB.transaction(
        function (transaction) {
            transaction.executeSql("SELECT * FROM configuracion;", [],

                function retrieveData(transaction, results) {

                    if(results.rows.length == 0)
                        reiniciar(false);

                    //Usage: results.rows.item(int)[columnName];
                    for(var i = 0; i < results.rows.length; i++) {
                        //This will return the data from row i (the integers from loop) from the column 'foobar'
                        configuracionLocal = results.rows.item(i)['configuracion_json'];

                        var confJSON = $.parseJSON(configuracionLocal);

                        localidad.ciudad = confJSON.ciudad;
                        localidad.comunidad = confJSON.comunidad;

                        reiniciar(true);

                        $("#localidadConf").val(confJSON.ciudad);
                        $("#comunidadConf").val(confJSON.comunidad);

                        var emisor = "";
                        var pluviometria = 0.0;

                        //ENCONTRAMOS EN JSON FABRICANTES EL EMISOR Y SU RENDIMIENTO CORRESPONDIENTE
                        $.each(fabricantes, function(i, item) {
                              //use obj.id and obj.name here, for example:
                              $.each(item.emisores, function(y, itemY) {
                                  //use obj.id and obj.name here, for example:
                                  if(itemY.codEmisor == confJSON.codEmisor){

                                      $.each(itemY.rendimientos, function(z, itemZ) {
                                          //use obj.id and obj.name here, for example:
                                          if(itemZ.codRendimiento == confJSON.codRendimiento){

                                              emisor = item.nombreFabricante+" / "+itemY.tipo+" "+itemY.modelo+" / "+itemZ.nombreTobera;
                                              pluviometria = itemZ.pluviometria;
                                          }
                                      });
                                  }
                              });
                        });

                $.post("buscarCondicionesHoyP", {localidad :$.toJSON(localidad)}, function(response) {

                     climaHoyJSON = response;

                     /*Caso devuelve error*/
                     if(climaHoyJSON.tipo != undefined){

                          error = true;
                     }else{
                         /*radiación solar incidente, convertida en mm/día*/
                          var Rs = calculaRs(climaHoyJSON.radiacion,climaHoyJSON.kt,climaHoyJSON.tempMax,climaHoyJSON.tempM

                          var ETCorregida = calculaET(Rs,climaHoyJSON.tempMedia,climaHoyJSON.precipitacion);

                          var tiempoRiego = calculaTiempoRiegoConfiguracion(ETCorregida,pluviometria);

                          var n = 4; //NUMERO DE CAMPOS TABLA PREVISION D
                          var tds = '<tr class="info">';
                          for(var i = 0; i < n; i++){
                                 switch(i)
                                 {
                                     //CASE 0 INDICAMOS EL DIA DE PREVISION
                                     case 0:
                                         tds += '<td>Hoy</td>';
                                         break;
                                     case 1:
                                         tds += '<td>'+climaHoyJSON.tempMedia+' (°C)</td>';
                                         break;
                                     case 2:
                                         tds += '<td>'+climaHoyJSON.precipitacion+' (mm)</td>';
                                         break;
                                     case 3:
                                         tds += '<td>'+tiempoRiego+'</td>';
                                         break;
                                 }
                          }
                          tds += '</tr>';
                          $("#tablaConfiguracion").append(tds);
                     }

                    /*PETICION PREVISION*/
                    $.post("buscarCondicionesSemanaP", {localidad :$.toJSON(localidad)}, function(response) {

                    previsionHoyJSON = response;
```

```
/*Caso devuelve error*/
if(previsionHoyJSON.tipo != undefined){

    error = true;
}else{

       var tMedias = $.parseJSON(previsionHoyJSON.tempMedia);
       var tMaximas = $.parseJSON(previsionHoyJSON.tempMax);
       var tMinimas = $.parseJSON(previsionHoyJSON.tempMin);
       var precipitaciones = $.parseJSON(previsionHoyJSON.precipitacion);
       var variasRadiaciones = false;

       if(previsionHoyJSON.radiacion2 != null){

           var radiacion = $.parseJSON(previsionHoyJSON.radiacion);
           var radiacion2 = $.parseJSON(previsionHoyJSON.radiacion2);
           variasRadiaciones = true;
       }

       for(var j = 0; j < 7; j++){ //7 DIAS-7FILAS

            var Rs = 0.0;
            var ET = 0.0;
            var tiempoRiego = "";
            /*CALCULO TIEMPO DE RIEGO*/
            if(variasRadiaciones){

                var dias = parseInt(radiacion.dias);
                var valor = parseInt(radiacion.valor);

                var valor2 = parseInt(radiacion2.valor);


                if(dias>=j+1)
                    Rs = calculaRs(valor,previsionHoyJSON.kT,parseFloat(tMaximas["dia_"+j]),parseFloat(tMinimas["dia_"+j]),parseFloat(tMedias["dia_"+j]));
                else
                    Rs = calculaRs(valor2,previsionHoyJSON.kT,parseFloat(tMaximas["dia_"+j]),parseFloat(tMinimas["dia_"+j]),parseFloat(tMedias["dia_"+j]));

            }else
                Rs = calculaRs(parseFloat(previsionHoyJSON.radiacion),previsionHoyJSON.kT,parseFloat(tMaximas["dia_"+j]),parseFloat(tMinimas["dia_"+j]),parseFloat(tMedias["dia_"+j]));

            if("dia_"+j in tMinimas && "dia_"+j in tMaximas && "dia_"+j in tMedias && "dia_"+j in precipitaciones){

                 ET = calculaET(Rs,parseFloat(tMedias["dia_"+j]),parseFloat(precipitaciones["dia_"+j]));
                 tiempoRiego = calculaTiempoRiegoConfiguracion(ET,pluviometria);

            }else
                 tiempoRiego="-";

            /*FIN CALCULO RIEGO*/

              var n = 4; //NUMERO DE CAMPOS TABLA PREVISION D
              var tds = '<tr class="warning">';
              for(var i = 0; i < n; i++){
                  switch(i)
                   {
                       //CASE 0 INDICAMOS EL DIA DE PREVISION
                       case 0:
                           switch(j){
                               case 0:
                                   tds += '<td>Mañana</td>';
                                   break;
                               case 1:
                                   tds += '<td>Pasado</td>';
                                   break;
                               case 2:
                                   tds += '<td>+3</td>';
                                   break;
                               case 3:
                                   tds += '<td>+4</td>';
                                   break;
                               case 4:
                                   tds += '<td>+5</td>';
                                   break;
                               case 5:
                                   tds += '<td>+6</td>';
                                   break;
                               case 6:
                                   tds += '<td>+7</td>';
                                   break;
                           }
                           break;
                       case 1:
                           tds += '<td>'+tMedias["dia_"+j]+' (ºC)</td>';
                           break;
                       case 2:
                           tds += '<td>'+precipitaciones["dia_"+j]+' (mm)</td>';
                           break;
                       case 3:
                           tds += '<td>'+tiempoRiego+'</td>';
                           break;
                   }
              }
              tds += '</tr>';
              $("#tablaConfiguracion").append(tds);
          }
       }
 },"json"); //fin post prevision
                       },"json"); //fin post Clima hoy
                       if(error){
                           $('#errorCargaConfiguracion').show();
                       }
                   }
               });
          }
     );
}
```

Como se puede observar, el método anterior posee cierta complejidad ya que en el momento de la consulta en caso de que haya configuraciones previas hay que realizar las

peticiones pertinentes al servidor para calcular el tiempo de riego óptimo para el día actual así como para realizar la previsión semanal.

Por otro lado, también se han implementado otros dos métodos para añadir y/o actualizar configuraciones:

```
function addConfiguracion(id,configuracion){

    try{

        SMDB.transaction(function (transaction) {
            transaction.executeSql("INSERT INTO configuracion(id, configuracion_json) VALUES (?,?)", [id,configuracion]);
        });

        selectConfiguracion(); //SETEA VARIABLE

    }catch(e) {

        console.log("Error al insertar configuracion: "+e+".");
    }

}

function updateConfiguracion(configuracion_json){

    try{

        SMDB.transaction(
            function (transaction) {
                transaction.executeSql("UPDATE configuracion SET configuracion_json=? WHERE id = 1", [configuracion_json]);
            }
        );

        selectConfiguracion(); //SETEA VARIABLE

    }catch(e) {

        console.log("Error al actualizar configuracion: "+e+".");
    }
}
```

CAPITULO 7. MANUAL DE USUARIO

En este punto se va abordar una guía de cara a facilitar el uso de la herramienta Smart Rain a un usuario final.

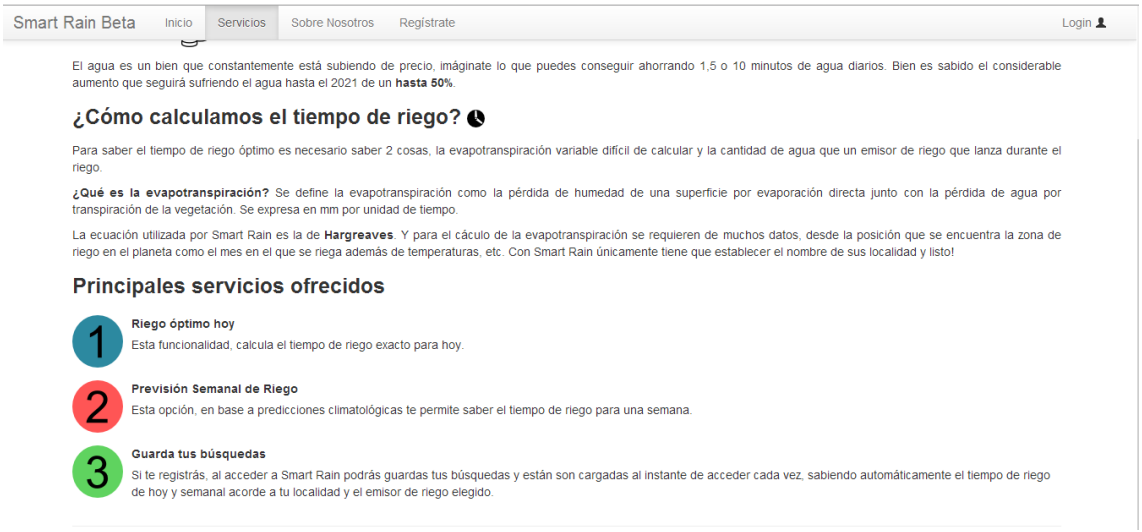
PORTADA

La primera pantalla que un nuevo usuario encontrará al conectarse a Smart Rain es una portada informativa donde se detallan desde los servicios ofrecidos como la posibilidad de comenzar a usar la herramienta o incluso la de registrase y usar la versión premium aprovechando todas sus ventajas.

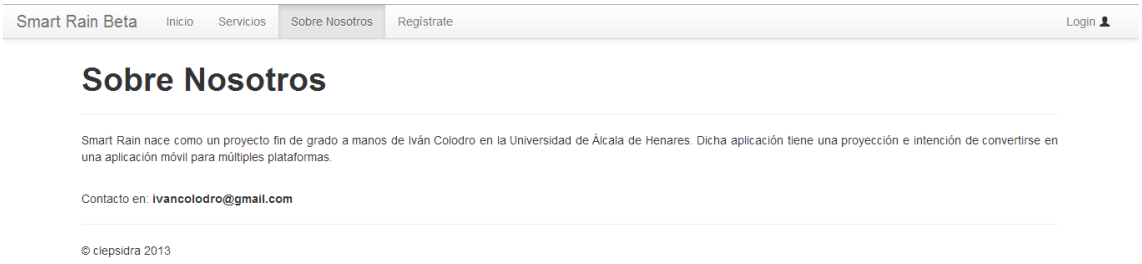
Portada/Inicio:

Desde inicio se puede acceder a cualquier página de la web, desde el uso de la herramienta, hasta leer los servicios ofrecidos o obtener información de contacto, etc.

Portada/Servicios



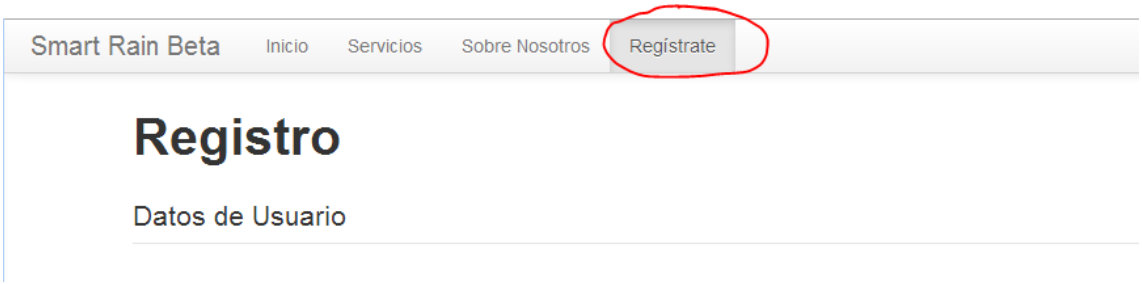
Portada/Sobre Nosotros



Portada/Registro

Siendo una funcionalidad importante el registro para captar usuarios premium, este se hace accesible de tres maneras, mediante el menú superior, mediante inicio así como en la pantalla de login.

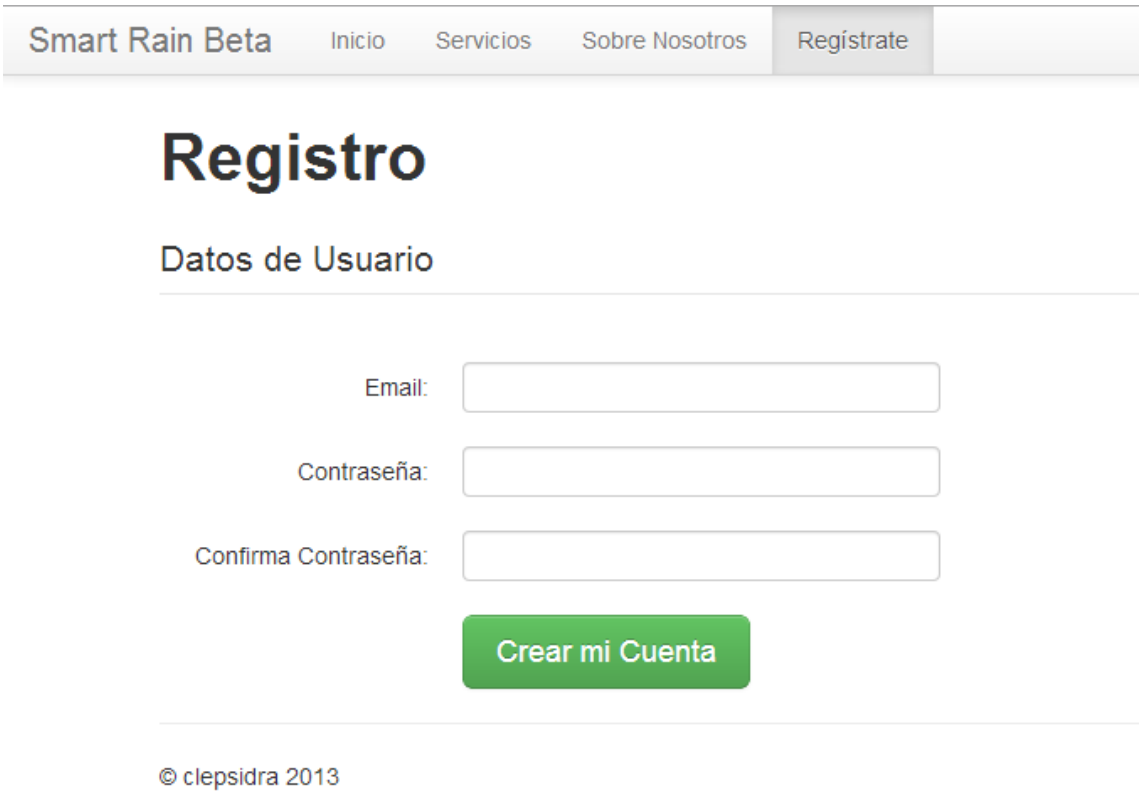
Mediante el menú superior



Mediante Inicio

Pantalla registro

Para registrarse el usuario debe de proporcional al sistema su correo electrónico así como una contraseña que le permita acceder a la versión premium.

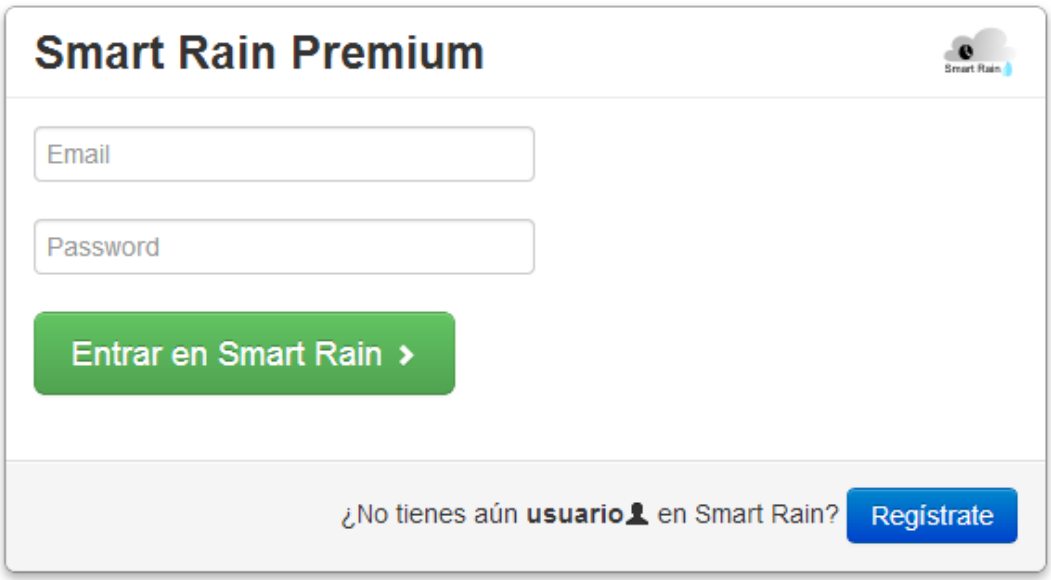


Portada/Login

Para acceder a la versión premium y por tanto a previamente a la pantalla de login, se puede realizar nuevamente mediante dos opciones. La primera , una vez realizado un registro exitosamente y por otro lado mediante la vista de inicio.

Acceso

Pantalla login



SMART RAIN VISITANTE

Esta opción de la herramienta Smart Rain ofrecida que no requiere el registro en la web es accesible bien mediante la URL … */smartrain/SMVisitante* ó mediante la portada, concretamente en inicio:

Una vez se haya pulsado el botón de acceso mostrado en la captura anterior, la pantalla mostrada es la siguiente:

Como se puede observar, las opciones de la herramienta son 3, el cálculo del tiempo de riego óptimo para "hoy", el cálculo del tiempo de riego óptimo para "durante la semana" y por último una sección de ayuda para aclarar algunas dudas al usuario de cara al funcionamiento del proceso de la consulta.

En las funcionalidades para calcular el tiempo de riego óptimo se hace un pequeño avance de la sección ayuda, de cara a informar al usuario, los datos que necesita ingresar en el sistema. Estas indicaciones se encuentran en el margen derecho:

Riego Hoy

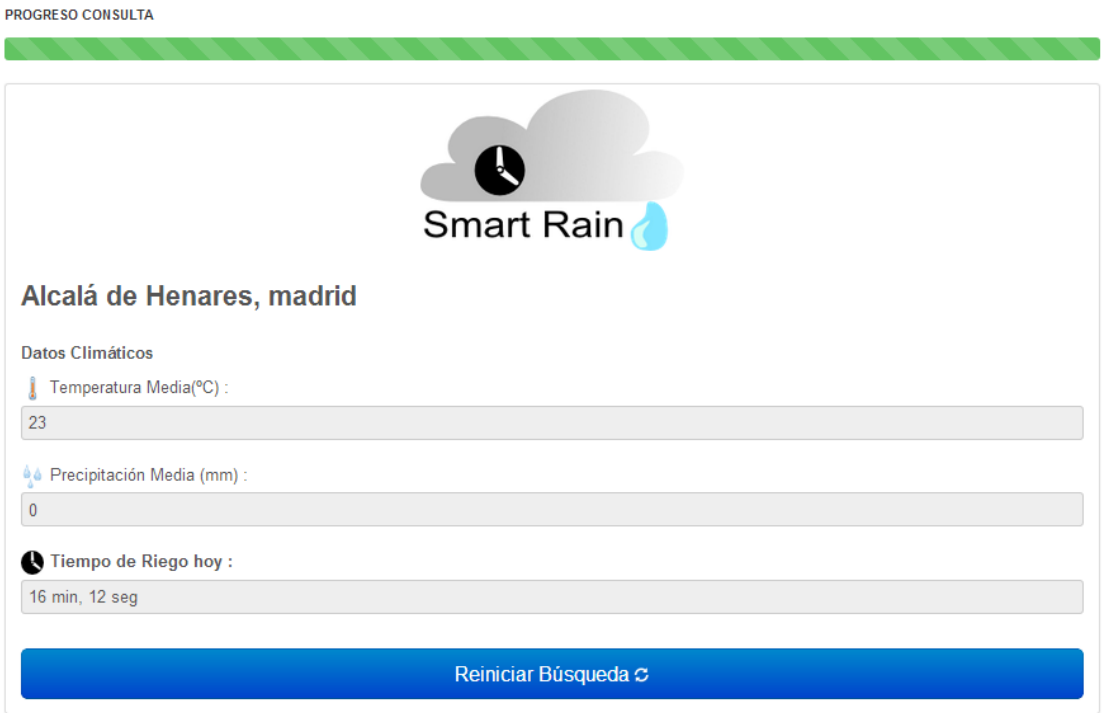
Para calcular el tiempo óptimo actual para una determina localidad, el usuario debe como primer paso insertar el nombre de la localidad así como la comunidad autónoma a la que pertenece tal y como se puede leer en las indicaciones básicas que aparecen en el margen.

Supongamos el caso de querer saber el tiempo óptimo de riego para la ciudad de Alcalá de Henares:

Una vez insertada la localidad, si todo va bien y el servidor no responde que la localidad introducida no puede ser encontrada (caso de que el usuario haya escrito mal la localidad a buscar), el siguiente paso, es elegir el tipo de emisor de riego que se está utilizando, imprescindible, ya que eligiendo el modelo, el sistema reconocerá que cantidad de agua suelta dicho emisor por hora, y de esta manera seguir con el cálculo del tiempo de riego óptimo.

Por el momento, el sistema posee varios modelos de los fabricantes Rain Bird y Hunter.

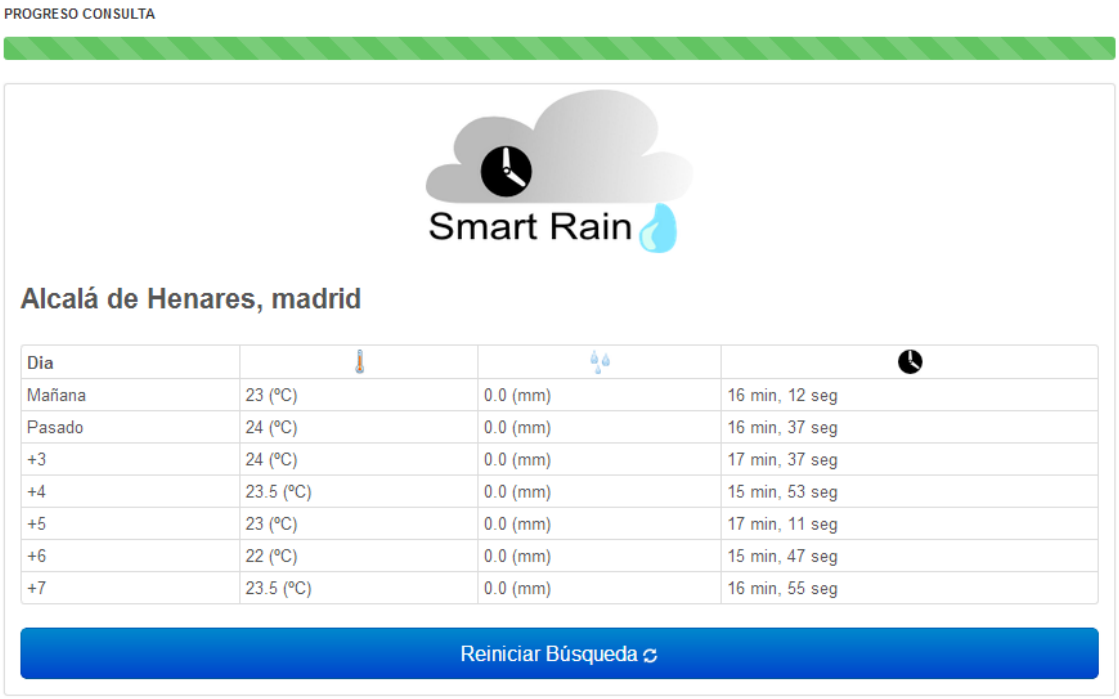
Una vez escogido, el fabricante, emisor y boquilla se le mostrará al usuario el tiempo de riego óptimo de la siguiente manera:



Para realizar una nueva búsqueda, simplemente hay que pulsar *Reiniciar Búsqueda.*

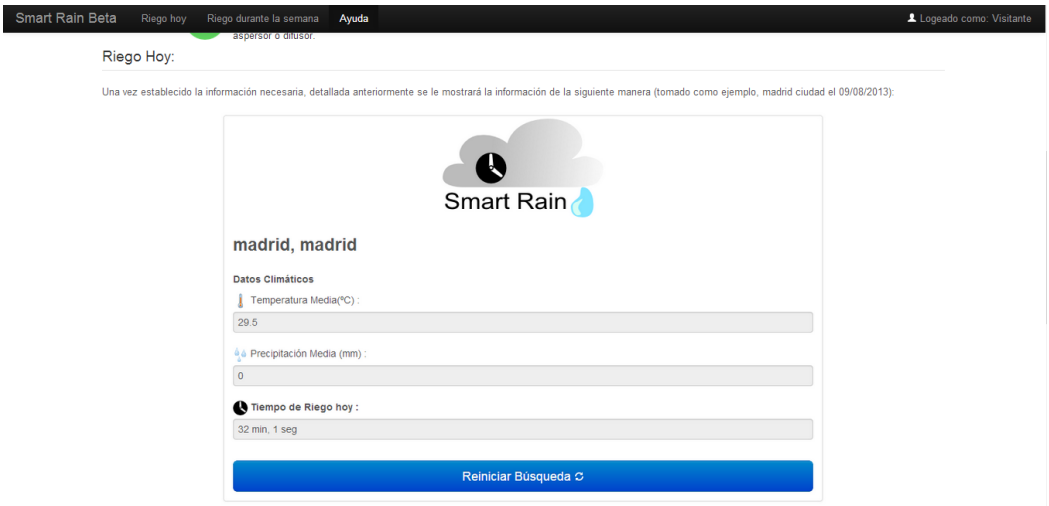
Riego durante la semana

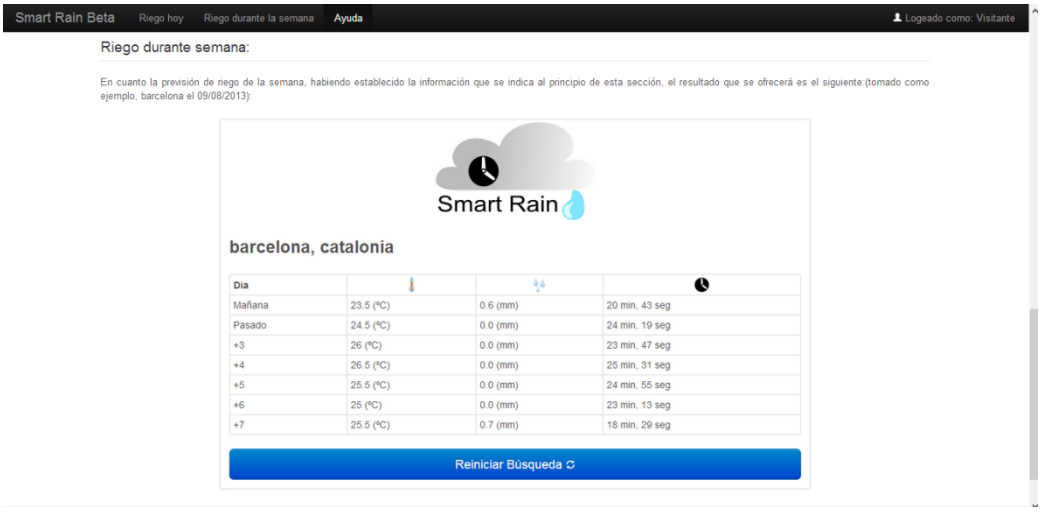
En este punto se omitirán los dos primeros pasos ya que son similares al del punto anterior. Por el contrario, y como es lógico, el resultado será diferente ya que habrá que mostrar una previsión para una semana. El resultado es como sigue:



Ayuda

En esta sección, únicamente se explica algo más detalladamente el funcionamiento de la herramienta con el objetivo de solucionar preguntas que puedan surgir al usuario, además se muestra algún ejemplo.

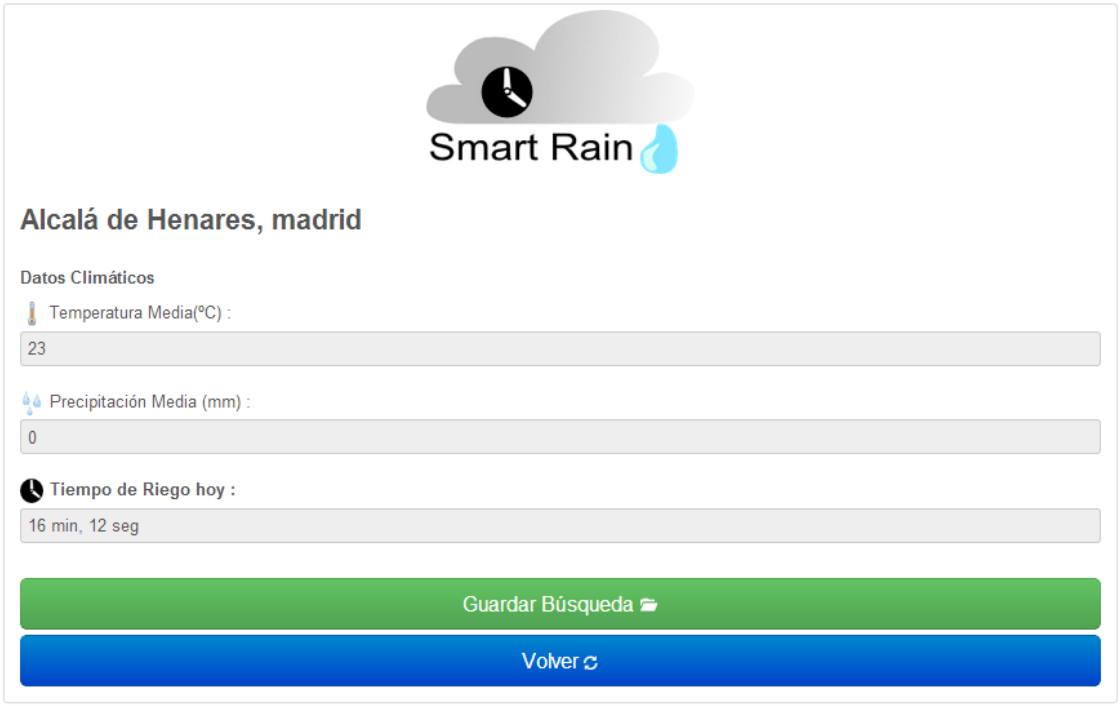




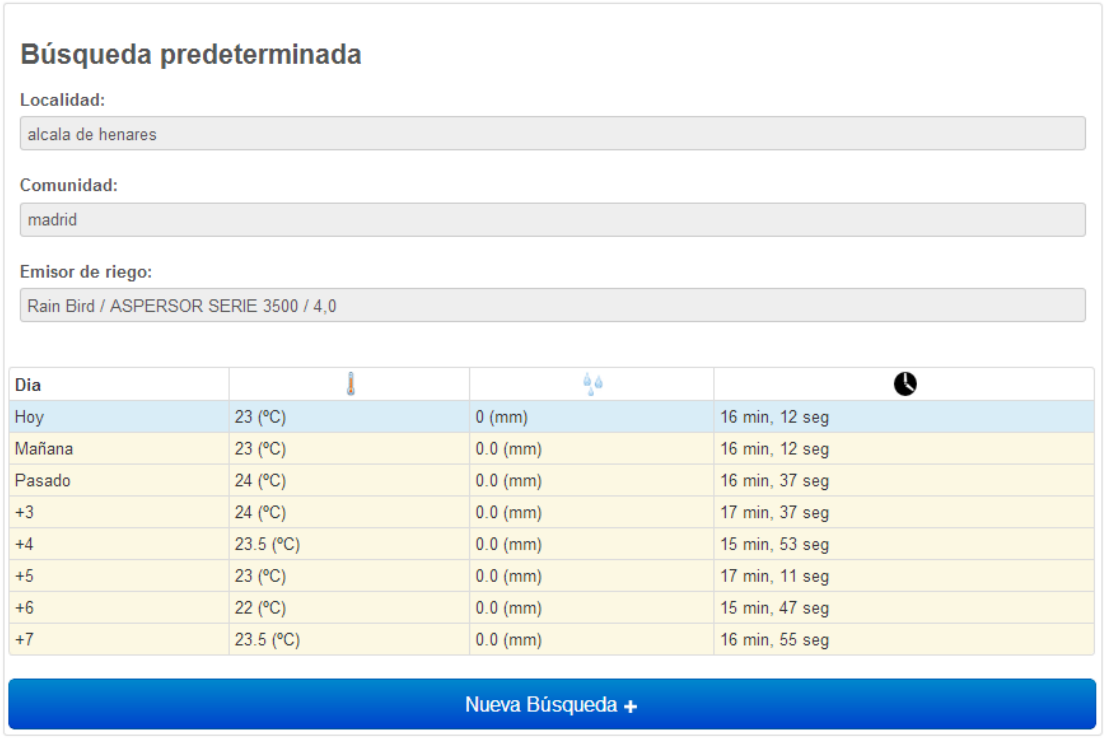
SMART RAIN PREMIUM

Por el momento, la versión premium de Smart Rain permite al usuario guardar sus búsquedas, para así, al conectarse, recibir automáticamente los datos de riego óptimo. Además de lo expuesto, otra de las diferencias respecto a la versión estándar reside en algunas indicaciones en la sección ayuda para utilizar las nuevas aplicaciones de la herramienta.

La metodología para realizar las búsquedas son exactamente iguales a las mostradas hasta ahora con la diferencia que en la pantalla del resultado, tanto de la funcionalidad *riego hoy* y *riego durante la semana* aparece un nuevo botón que permite guardar la búsqueda tanto para la localidad como para el emisor de riego y así calcular automáticamente los tiempos de riego automáticamente cada vez que el usuario se conecte.



Una vez guardada la búsqueda, al conectarse a Smart Rain, el usuario obtendrá los datos automáticamente de la siguiente manera:



CAPITULO 8. LÍNEAS FUTURAS, COMERCIALIZACIÓN.

El futuro esperado para Smart Rain es su evolución a una **aplicación móvil** debido a que se encuentra un nicho de mercado más atractivo a la hora de conseguir beneficios. A priori las plataformas deseadas dónde se planea lanzar la aplicación son Android e IOS.

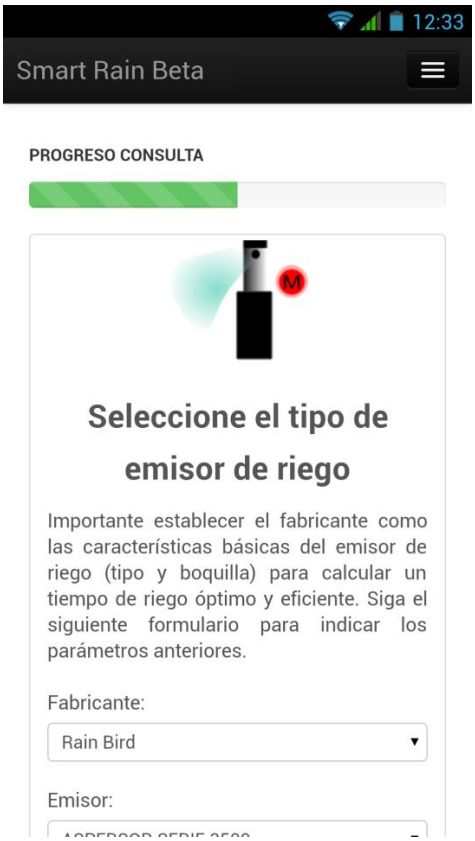
Los planes comentados, ciertamente, cambian algo la arquitectura de la aplicación, no obstante, se han cuidado temas de diseño de las vistas, tal y como se anunció en el anteproyecto. Por tanto, las vistas desarrolladas, combinado HTML y CSS, hacen que Smart Rain actualmente se asemeje a una aplicación tanto para un dispositivo móvil como para una Tablet.

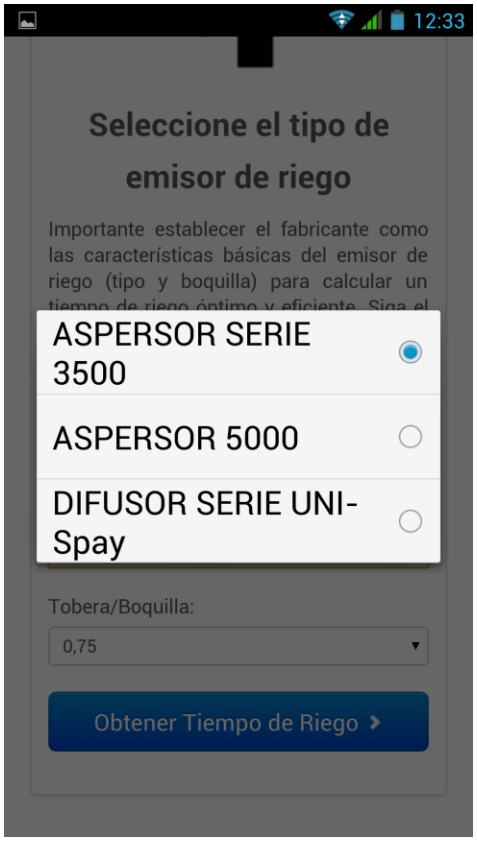
VISTAS DE SMART RAIN EN UN DISPOSITIVO MÓVIL

El dispositivo que se ha utilizado para probar Smart Rain es un *Tianhe H920.*

A continuación se mostrarán varias vistas de la herramienta de Smart Rain en dicho dispositivo.


12:26
192.168.1.59:8080/smartrain/SM
Smart Rain Beta
Riego hoy
Riego durante la semana
Ayuda
PROGRESO CONSULTA
Smart Rain
Hoy
Busca tu pueblo o ciudad


12:33
Smart Rain Beta
PROGRESO CONSULTA
Seleccione el tipo de emisor de riego
Importante establecer el fabricante como las características básicas del emisor de riego (tipo y boquilla) para calcular un tiempo de riego óptimo y eficiente. Siga el siguiente formulario para indicar los parámetros anteriores.
Fabricante:
Rain Bird
Emisor:

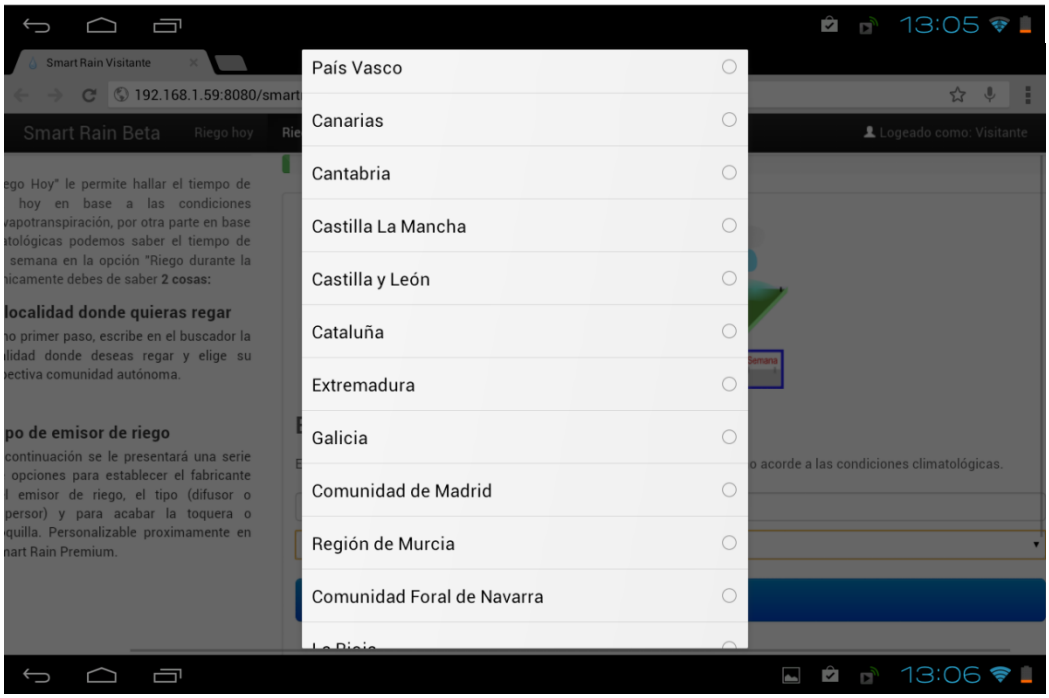

12:33
Seleccione el tipo de emisor de riego
Importante establecer el fabricante como las características básicas del emisor de riego (tipo y boquilla) para calcular un
ASPERSOR SERIE 3500
ASPERSOR 5000
DIFUSOR SERIE UNI-Spay
Tobera/Boquilla:
0,75
Obtener Tiempo de Riego

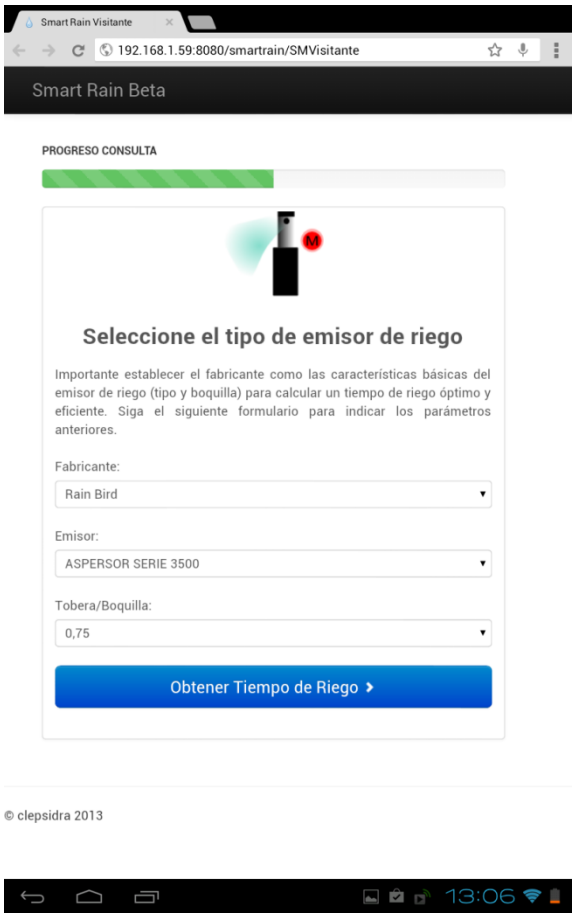

12:34
Smart Rain
Alcalá de Henares, madrid
Datos Climáticos
Temperatura Media(ºC) :
24
Precipitación Media (mm) :
0
Tiempo de Riego hoy :
16 min, 37 seg
Reiniciar Búsqueda

VISTAS SMART RAIN EN UNA TABLET

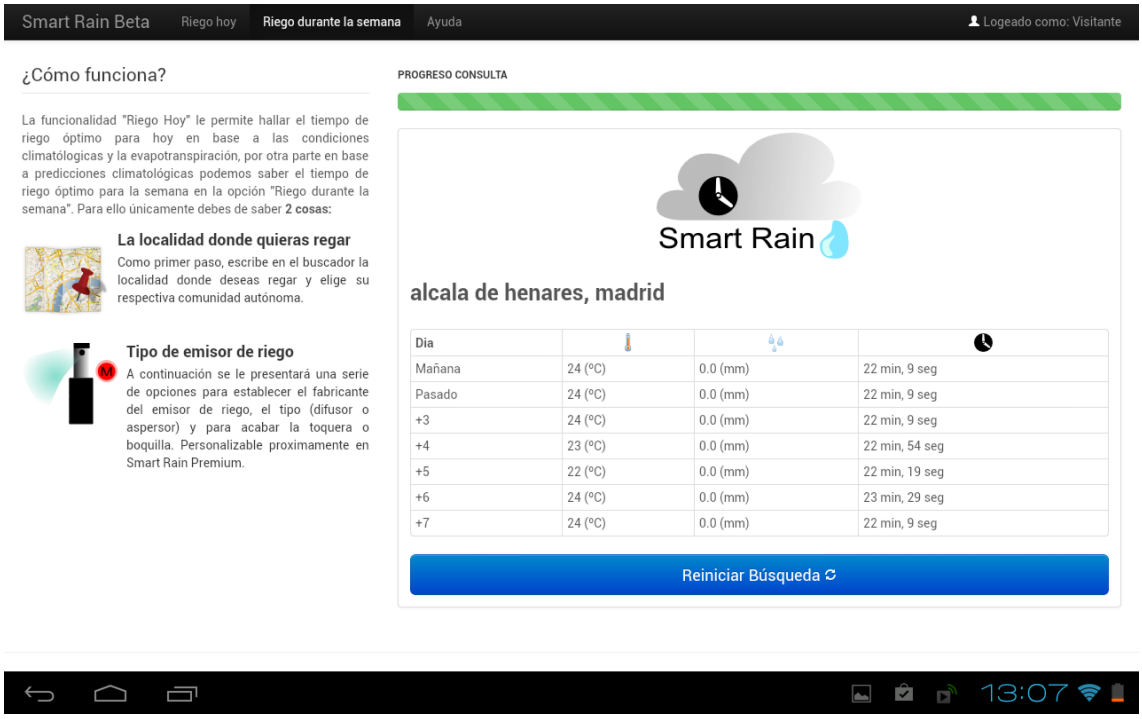
El dispositivo que se ha utilizado para probar Smart Rain es un Acer A500

A continuación se mostrarán varias vistas de la herramienta de Smart Rain en dicho dispositivo.



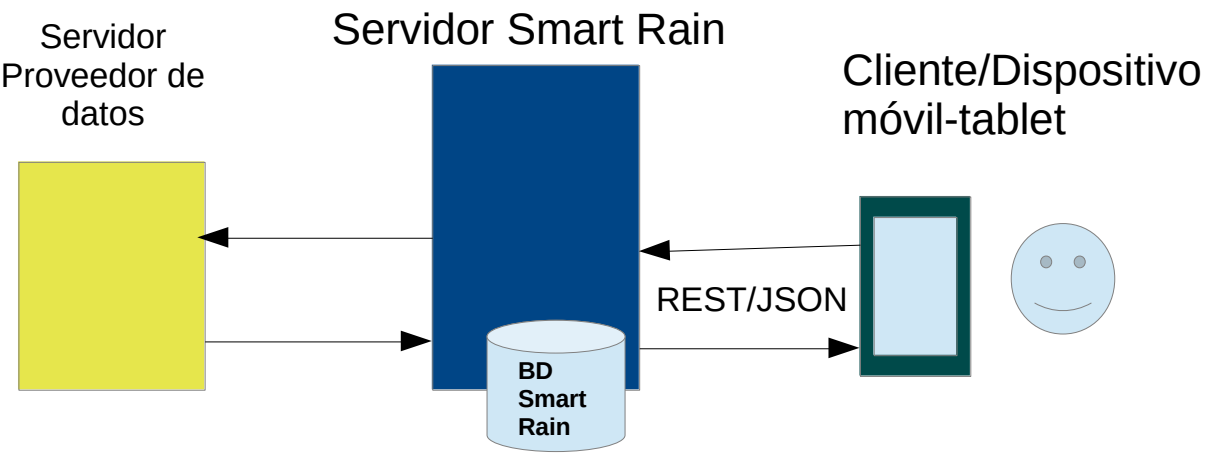


(Posición vertical del dispositivo)



LA NUEVA ARQUITECTURA

La intención final de convertir Smart Rain a una aplicación para los sistemas operativos de Android e IOS hace que la arquitectura tenga que ser cambiada. No obstante, la nueva arquitectura no varía tanto de la actual.



Servidor

Por un lado, la parte del servidor de la actual aplicación web se convertirá en un servicio web, ya que se requerirá el intercambio de datos entre dos aplicaciones diferentes.

Por otro lado, destacar el uso de REST frente a la tradicional arquitectura SOAP para la implementación de un servicio web. En los últimos años, la transferencia de estados representacionales o lo que es los mismo, REST, se ha convertido en una alternativa muy popular centrada en la información y opuesta a los servicios web basados en SOAP. Además, el framework utilizado para desarrollar Smart Rain e implementar el patrón de diseño MVC, Spring 3, incluye la compatibilidad con el modelo REST lo que facilita la conversión de lo realizado de forma simple a un servicio web.

Pero, ¿Qué es en concreto REST?

Analizando, su significado al español, *transferencia de estado representacional,* es posible sacar en claro que es REST.

- Transferencia: REST implica la transferencia de datos de recursos, en forma de representación, de una aplicación a otra.

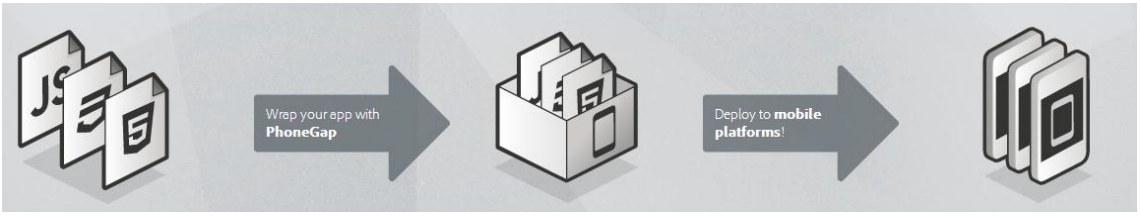
- Estado: cuando se trabaja con REST, preocupa más el estado de un recurso que las acciones que se pueda realizar con éstos.

- Representacional: los recurso de REST pueden representarse en prácticamente cualquier formato, XML, JSON o HTML.

Cliente

A priori, parece que el lado del cliente será la parte que más cambios puede sufrir respecto a la aplicación actual ya que se requiere el desarrollo de aplicaciones para Android e IOS. Sin embargo, se planea utilizar el framework **PhoneGap**.

PhoneGap es un framework de código abierto y gratuito que permite el desarrollo de aplicaciones móviles a partir de la tecnología de la web como pueda ser HTML, CSS y javascript.



Una de las grandes ventajas que aporta este framework es la característica de poder desarrollar una aplicación para cualquier sistema operativo de dispositivos móviles.

Supported Features

The chart below shows which APIs are available for each device. Read more about them in our Phonegap Docs.

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+	WebOS	Windows Phone 7 + 8	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	x	✓	✓	x	x	✓	✓	x	✓
Contacts	✓	✓	✓	✓	✓	x	✓	✓	✓
File	✓	✓	✓	✓	✓	x	✓	x	x
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	x	x	x	✓	x	x
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	x	x

FUENTES DE INGRESOS

Este proyecto persigue en esencia tres fuentes de ingreses.

En primer lugar, destacar, que la línea a seguir es el desarrollo de dos versiones de la aplicación. Una gratuita que se base en la obtención de ingresos mediante la publicidad y otra versión premium, a la que se establecerá un precio, y se añadirán más funcionalidades, tales como las que ya están desarrolladas que permiten guardar las búsquedas así como dar la posibilidad de personalizar el emisor de riego utilizado.

Hasta aquí se ha dado a entender dos de las tres fuentes de ingresos, la tercera consiste en la negociación con los fabricantes de emisores de riego para que incluyan sus modelos en la aplicación y que esto les sirva como un portal publicitario, así pues se negociaría una cuota anual acorde a los modelos que incluyan en Smart Rain.

Por tanto, en resumen, las fuentes de ingresos son:

- Publicidad
- Ventas de la versión Premium
- Fabricantes

LOS PROVEEDORES DE DATOS CLIMATOLÓGICOS

El proveedor de datos escogido finalmente ha sido World Weather. Sin embargo, una de las debilidades de Smart Rain es la dependencia de estos proveedores ya que sin los datos ofrecidos por los mismos, la herramienta no es capaz de aportar valor al usuario final.

Ante el riesgo que pudiera suponer una caída de los servidores de World Weather, aunque improbable por ser una empresa seria, pero no imposible, no se descarta el desarrollo de conexiones a otros servidores alternativos como puedan ser Open Weather o Wunderground que funcionen de manera complementaria en caso de la presencia de irregularidades con los servidores de World Weather.

Por otro lado, destacar, que normalmente los proveedores de datos climatológicos ofrecen servicios premium. Este es el caso de World Weather. La cuota por un servicio premium será valorado y en todo caso negociado según la evolución en el mercado de la aplicación.

CONTRATACIÓN DE UN SERVIDOR

Para la puesta en funcionamiento de la herramienta Smart Rain es necesario la contratación de uno o varios servidores que satisfagan las necesidades de suministro de datos por parte del servicio web Smart Rain a los diferentes clientes.

La opciones planteadas por el presente autor son variadas, existen muchas soluciones para alojar un sitio web, desde comprar y crear un propio centro de datos, contratarlo a un de centro con una oferta de servidores dedicados o gestionados, o utilizar soluciones basadas en la nube. Sin importar la opción a elegir, la empresas deben de garantizar que su infraestructura de alojamiento maximice la seguridad, la fiabilidad y el rendimiento. El coste (los costes preliminares y los gastos vigentes) y la flexibilidad también son consideraciones importantes para las empresas de cualquier tamaño.

A priori, la solución de crear un propio centro de datos queda descartada debido a la alta inversión que requiere en un primer momento. Por tanto las opciones barajadas son contratar un servidor dedicado ó utilizar una solución basada en la nube ya que son las opciones más válidas para empresas que desean reservar capital inicialmente.

Por un lado, Amazon es una de las opciones a elegir. Amazon Web Services (AWS) proporciona recursos informáticos y de almacenamiento fiables, escalables y rentables en los que alojar aplicaciones web. Además ofrece varios componentes a combinar según las necesidades requeridas, entre los que destacan:

- Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 ofrece capacidad informática de tamaño modificable en la nube. El usuario podrá definir su entorno Amazon EC2 virtual con el sistema operativo, los servicios, las bases de datos y la pila de plataforma de aplicaciones que necesite su aplicación alojada. Amazon EC2 ofrece una consola de gestión completa y API para la gestión de sus recursos informáticos.

- Amazon Simple Storage Service (Amazon S3). Amazon S3 proporciona una sencilla interfaz de servicios web que permite almacenar y recuperar la cantidad de datos que se desee, cuando se desee, y desde cualquier parte de la Web. Ofrece seguridad, una alta disponibilidad y una gran duración. Amazon S3 almacena varias copias redundantes de los datos.

- Amazon Relational Database Service (Amazon RDS). Amazon RDS es un servicio web que facilita las tareas de configuración, utilización y escalado de una base de datos relacional en la nube. Proporciona una capacidad de base de datos rentable y de tamaño variable, a la vez que gestiona las tareas de administración de base de datos más lentas.

- Amazon SimpleDB. Amazon SimpleDB proporciona las funciones de base de datos básicas de indexado y realización de consultas. Podrá escribir sus aplicaciones para que hagan uso de la sencillez de Amazon SimpleDB y de su capacidad de escalar de forma transparente.

- Amazon CloudFront. Amazon CloudFront proporciona un sistema de entrega de contenido de alto rendimiento distribuido a nivel internacional. Su aplicación puede utilizar Amazon CloudFront para distribuir o transmitir contenido fácilmente a los usuarios con una baja latencia, grandes velocidades de transferencia de datos, sin compromisos y con una integración continua con Amazon S3.

- Amazon Simple Queue Service (Amazon SQS). Amazon SQS proporciona un sistema de colas seguro y de alto rendimiento para su aplicación, que le permite distribuir de forma fiable el trabajo entre los procesos de un sitio web.

Por otro lado, existe la opción de la contratación de un servidor dedicado por ejemplo en OHV. Posiblemente esta opción requiera de más conocimientos para la puesta en marcha de la web. No obstante, la configuración, si cabe, estarán más ajustadas y/o personalizadas a las necesidades de Smart Rain. Por contra, los primeros meses resultaría más caro la contratación de un alojamiento web en Amazon.

La contratación de un servidor dedicado en el sitio mencionado además permite la escalabilidad. Si se requiriese, se podría cambiar de máquina para así aumentar su rendimiento acorde a una subida usuarios de Smart Rain y poder satisfacer todas las peticiones realizadas de una manera eficiente.

Por tanto, a la hora de contratar un servidor dedicado se pensaría en una máquina potente pero a la vez modesta para un empiece como pueda ser el servidor dedicado de OHV KS 24G con las siguientes características:

Como se puede observar, el precio mensual sería de 37.50€. Sí dicho servidor se quedase corto de prestaciones debido a una mayor demanda por un aumento de los usuarios, se podría cambiar a otro más ponte sin mayor problema.

Una vez, comentada las dos opciones barajadas, se optará finalmente por la contratación de un servidor dedicado en OHV, básicamente por dos razones. Por un lado la configuración y total personalización del servidor acorde a las necesidades de Smart Rain, así como la representación de una cuota fija mensual que facilita un análisis económico, ya que por contra Amazon, generalmente cobra acorde a el uso de la red, lo que está muy bien y puede significar un ahorro de costes pero representaría un coste variable y haría más difícil el análisis económico que se realizará en el capítulo 9.

CAPITULO 9. ANÁLISIS ECONÓMICO

En este capítulo se abordará un análisis económico, revisando las inversiones necesarias para llevar a cabo el proyecto de Smart Rain. Se elaborará un presupuesto a largo plazo indicando tanto gastos e ingresos esperados, teniendo en cuenta que el desarrollo para este año ya está realizado así como lo que faltaría por realizar de cara a una comercialización. Además se incluirán todos los gastos necesarios como mantenimiento de un servidor ó licencias requeridas. Finalmente, mediante herramientas financieras, se sacarán conclusiones sobre la rentabilidad y fiabilidad que tiene Smart Rain como un proyecto de inversión.

ELABORACIÓN DEL PRESUPUESTO

La presupuestación es una herramienta en manos de la planificación para la consecución de los objetivos empresariales.

Antes de la elaboración del presupuesto general a largo plazo, es necesario determinar otros presupuestos complementarios como el de producción así como el presupuesto de ventas.

Presupuesto de producción

Los presupuestos de producción son estimados acorde a los procesos/ materiales necesarios para la fabricación unitaria de un producto. Este proyecto, al ser de carácter informático, las únicas labores directamente relacionas con la producción están relacionadas con la mano de obra dada por las tareas de análisis/diseño e implementación. Por tanto únicamente se tendrá en cuenta el importe del desarrollo total de Smart Rain acorde a los dos roles descritos, **sin hacer un desglose de costes fijos y variables** y además como peculiaridad se añadirá otros costes típicos de un presupuesto no productivo como son las **cotizaciones a la seguridad social** ya que en el presupuesto general los datos utilizados ya tienen en cuenta las obligaciones de impuestos con el estado.

Previo análisis con una herramienta de planificación de proyectos como COCOMO II, la estimación concluida por horas a realizar por rol son las siguientes:

- Análisis y Diseño = 4.5 Meses/persona = 792 Horas/Persona
- Implementación y Pruebas = 5 Meses/persona = 880 Horas/Persona

Por otro lado es necesario considerar el coste mensual por un analista/diseñador , que es de 2.333 € y el coste mensual por un programador es de 1.750 €. La cotización a la Seguridad Social implica un coste a sumar por el empresario del 29,9 % en representación de:

- Contingencias comunes = 23.60
- Desempleo = 5.5
- Fogasa = 0.20
- FP = 0.6

concepto	Importe	Seguridad Social	Coste total	Coste por hora
PERSONAL/ROLES				
Analista/diseñador	10499	3139,20	13638,20	17,22
Programador	8750	2616,25	11366,25	12,92
		TOTAL COSTES	25004,45 €	30,14 €

Presupuesto de ventas

Este apartado consiste en el establecimiento del plan de ventas o ingresos para el periodo presupuestario establecido dentro del plan estratégico de la empresa. En este caso, recordar que los planes estratégicos son elaborados para un periodo de entre 3 a 5 años. Acorde al presupuesto general, tendremos en cuenta un periodo de 5 años lo que permitirá un posterior análisis financiero sobre la rentabilidad económica de Smart Rain.

La línea de productos a sacar en el mercado, a priori, son dos, la versión gratuita de Smart Rain que basa sus ingresos en la publicidad y por otro lado, Smart Rain premium con un precio unitario de 2,99 € al usuario final, sin embargo, las plataformas de Google Play y Apple Store se quedan con un 30% del precio establecido, por tanto consideraremos el precio unitario de dicha versión de 2,093 €.

	Smart Rain free Version			
	Previsión Ventas	Precio Unitario	Ingresos por ventas	Ingresos por publicidad
2013	0	0	0	0
2014	500	0	0	50
2015	5000	0	0	500
2016	25000	0	0	2500
2017	50000	0	0	5000
TOTAL	8.050 €			

	Smart Rain Premium			
	Previsión Ventas	Precio Unitario	Ingresos por ventas	Ingresos por publicidad
2013	0	2,093	0	0
2014	100	2,093	209,3	0
2015	2000	2,093	4186	0
2016	10000	2,093	20930	0
2017	25000	2,093	52325	0
TOTAL	77.650 €			

Presupuesto general

Como consideración inicial, indicar y recordar que el requerimiento de producción para el producto final de Smart Rain es la disposición de un programador y un analista/diseñador a jornada completa durante cinco y cuatro meses y medio respectivamente por rol. Dichas funciones realmente están siendo desempeñadas por una sola persona y aunque ya la mitad del desarrollo está totalmente completado para la versión comercial de Smart Rain, la limitación de ser una persona en vez de dos requiere una ajuste temporal y trasladar tareas de producción al 2014 para así poder ser totalmente realistas.

Por tanto, del primer presupuesto, es decir, el de producción, concluimos lo siguiente: un coste de 10499 € por labores de análisis y diseño, y 8750€ por un programador. Alargada la fase productiva a 2014, dividiremos esas cantidades por dos para repartir la producción en 2 años tal y como se podrá ver en el presupuesto general.

	Año				
Concepto	2013	2014	2015	2016	2017
GASTOS	9624,5	11327,13	3868,88	3868,88	3868,88
Análisis/diseño	5249,5	5249,5	0	0	0
Implementación	4375	4375	0	0	0
Mantenimiento	0	1422,3	3418,88	3418,88	3418,88
Licencia Google	0	18,72	0	0	0
Licencia Apple	0	74,11	0	0	0
Servidor	0	187,5	450	450	450
INGRESOS	0	759,3	5436	24430	60325
publicidad	0	50	500	2500	6000
ventas directas	0	209,3	4186	20930	53325
sponsors/fabricantes	0	500	750	1000	1000
BENEFICIO ACUMULADO	-9624,5€	-20192,33€	-18625,21€	1935,91€	58392,03€

VALORACIÓN DE SMART RAIN COMO UN PROYECTO DE INVERSIÓN

A partir del presupuesto general del punto anterior, es posible la realización de diferentes gráficos y especificación de algunos conceptos financieros para determinar la viabilidad del proyecto Smart Rain, o lo que es lo mismo, su rentabilidad económica.

Las herramientas financieras a utilizar para valorar la rentabilidad económica de Smart Rain como un proyecto de inversión son las siguientes:

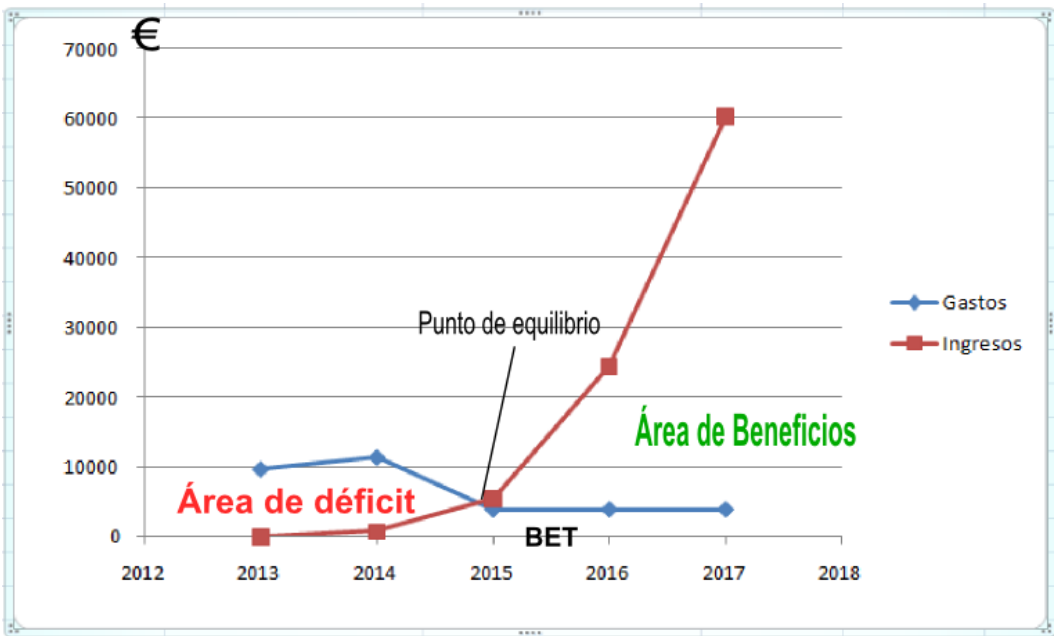
- Período de recuperación de inversión y punto de equilibrio
- ROI: beneficio acumulado
- Valor actual neto (VAN)

Período de recuperación de la inversión y punto de equilibrio

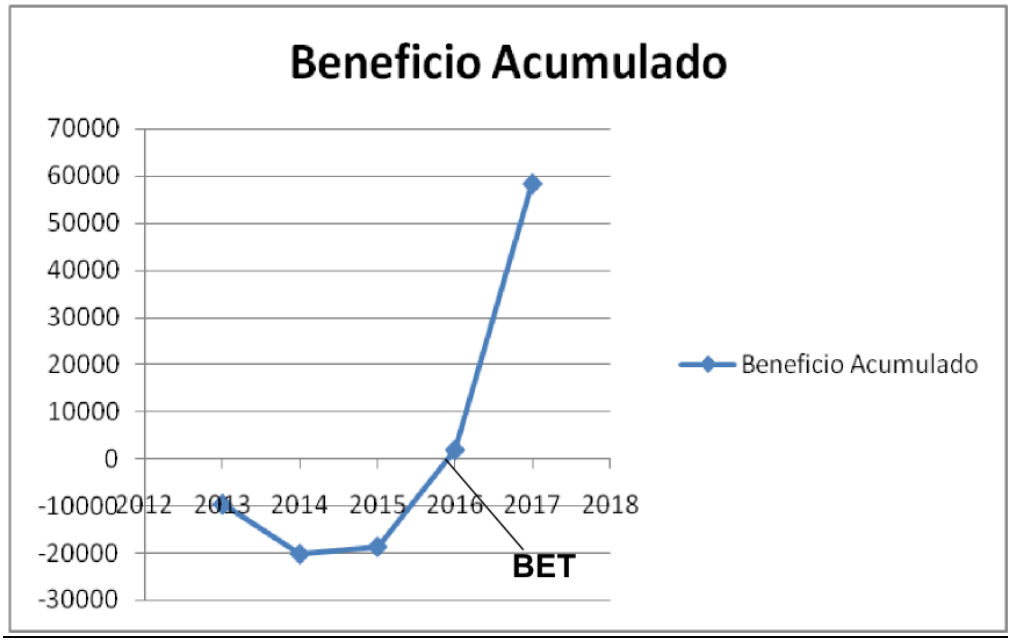
Como el propio nombre indica, el período de recuperación de la inversión, es el momento en el que se empieza a generar beneficio, concepto estrechamente relacionado con el punto de equilibrio donde los ingresos igualan a los gastos.

Obsérvense los siguientes gráficos de "Ingresos y Gastos", y "Beneficio Acumulado" para poder aprecian donde está el punto de equilibrio y por tanto determinar **cuando** en los 5 años planificados se empieza a generar beneficios, superando así los ingresos a los gastos.

Ingresos y Gastos



Beneficio Acumulado



ROI

El ROI, el retorno de la inversión como sus propias siglas indican ó también conocido como el beneficio calculado, es fácilmente apreciable en la planificación dada a 5 años tanto en el gráfico anterior así como en el presupuesto general:

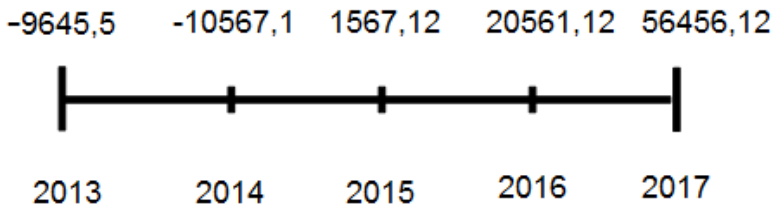
BENEFICIO ACUMULADO	-9624,5€	-20192,33€	-18625,21€	1935,91€	58392,03€

VAN

VAN, Valor Actual Neto o también conocido como Beneficio Total Actualizado. Se llama así porque es una medida financiera del Beneficio Actualizado adicional calculado bajo un tipo de interés determinado.

El interés a considerar será próximo al de mercado, que actualmente ronda sobre el 9%.

La siguiente línea temporal representa las cuantías previstas tanto necesarias como generadas por el proyecto(diferencia entre ingresos y gastos por año).



Por tanto, teniendo en cuenta que :

$$VAN(i) = \sum_{s=1}^{n} C_s \cdot (1+i)^{-s} - \sum_{s=0}^{n} D_s \cdot (1+i)^{-s}$$

Que quiere decir que el VAN es igual al valor actual de los cobros menos el valor actual de los pagos, ya aplicada la diferencia en la línea temporal representada, lo que hay que hacer es llevarnos los valores representados al momento actual.

VAN(0,09) = -9645,5 (1+0,09)^0 - 10567,1(1+0,09)^- 1+1567,12(1+0,09)^-2+20561,12(1+0,09)^-3 + 56456,12(1+0,09)^-4

VAN(0,09) = 37.850.82 €

Con este último dato podemos **afirmar la rentabilidad económica del proyecto de Smart Rain,** ya que sí el VAN > 0, indica que la inversión permite recuperar lo invertido haciendo frente al pago de todos los gastos derivados del proyecto, obteniendo además un beneficio adicional.

CAPITULO 10. CONCLUSIONES

Hoy en día el agua es un recurso escaso y cada vez más caro en nuestro país, he aquí, por tanto, la idea de una aplicación dirigida a usuarios finales, no expertos, con la que puedan hacer un uso eficiente del agua y la facilidad de tener un tiempo óptimo de riego al instante en dos simples pasos. Todo ello supone un gran ahorro para los bolsillos de los consumidores, ya que el agua no es un bien precisamente barato y como se ha mencionado, constantemente está aumentando el precio, por lo que un simple ahorro de 5 minutos diarios de media puede significar salvar una gran cantidad de dinero anualmente, además una manera indirecta, se adquiere un compromiso con el medio ambiente.

Aunque Smart Rain persigue un público no experto en materias de gestión de riegos y medio ambiente con el fin de solucionar labores cotidianas como el riego de un jardín, la herramienta también es útil para técnicos o jardineros profesionales que tengan a su cargo la gestión de parques, ya que Smart Rain ofrece unos datos orientativos para los mismos de gran utilidad, lo que puede incluso, conllevar el ahorro a los ayuntamientos que financian el mantenimiento de esos parques.

A lo que tecnología se refiere, se ha mostrado la utilización de un framework muy importante del Back-end, como es Spring, consiguiendo un diseño claro, separando la lógica de negocio de las vistas debido a la implementación del patrón Modelo-Vista-Controlador. Además, dicho framework, permite una complementación muy buena con otras tecnologías dando soluciones a muchos de los problemas que puedan surgir en el desarrollo de una aplicación, principal ventaja por la que se ha optado en vez de utilizar otros frameworks como Struts. Por parte del Front-end, destacar que se ha conseguido una óptima visualización responsiva que se adapta a los diferentes dispositivos que requieran las vistas de la web, ya sean móviles o tablets, y todo ello gracias a la utilización del framework Boopstrap, basado en el uso de Hojas de estilo y HTML5, facilitando una evolución a una aplicación móvil en un futuro próximo, dónde se encuentra un nicho de mercado mucho más atractivo a la hora de realizar ventas, sobre todo de cara, a la venta de un servicio como el ofrecido por Smart Rain que va dirigido a usuarios finales.

Finalmente, una vez expuesta la idea, las líneas de futuro a seguir, dónde, cómo y a quiénes vender la aplicación y concluyendo la gran oportunidad de negocio que supone, se ha llevado a cabo un análisis económico determinando la viabilidad del proyecto, verificando así el potencial de Smart Rain. Los resultados, realizando un previsión de ventas asequible, ha sido que se puede perfectamente asumir los gastos asociados a una puesta en marcha del sistema, además de la proyección y tendencia a no solo recuperar la inversión que se realice por la producción total de la aplicación sino que es también una apuesta casi segura a la obtención de beneficios.

CAPITULO 11. BIBLIOGRAFÍA

- David Sawyer McFarland (2012) . JavaScript y jQuery. Editorial Anaya
- Donald Brown, Chad Michael Davis y Scott Stanlick (2009). Struts 2. Editorial Anaya
- Craig Walls(2012). Spring. Editorial Anaya
- Hargreaves,G.H., Samani, Z.A.,(1985). Reference crop evapotranspiration from temperature.
- Samani, Z (2000).- Estimating Solar Radiation and Evapotranspiration Using Minimum Climatological Data. *Journal of Irrigation and Drainage Engineering*, Vol. 126, No. 4, pp. 265-267
- Hibernate - JBoss Community (2013) (Online) Disponible en: http://www.hibernate.org/
- Documentation | SpringSource.org(2013) (Online) Disponible en: http://www.springsource.org/documentation
- PostgreSQL: Documentation(2013) (Online) Disponible en: http://www.postgresql.org/docs/
- Bootstrap (2013) (Online) Disponible en: http://twitter.github.io/bootstrap/
- World Weather Online - Documentation (2013) (Online)Disponible en: http://developer.worldweatheronline.com/documentation
- JSON (2013) (Online) Disponible: http://www.json.org/
- Tema 5-Métodos de valoración de proyectos de inversión. Análisis y Valoración de Proyectos de Inversión. Segundo Curso. Grado Sistemas de Información. Universidad Alcalá de Henares.
- Presupuestos. Sistemas de Información Contable. Segundo Curso. Grado Sistemas de Información. Universidad Alcalá de Henares.
- Tema 1-Viabilidad de Proyectos. Gestión de Proyectos. Tercer Curso. Grado Sistemas de Información. Universidad Alcalá de Henares.